```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

FILEID**IOCIOPOST

```
  IIIIII      OOOOOO      CCCCCCCC    IIIIII      OOOOOO    PPPPPPPP      OOOOOO      SSSSSSSS  TTTTTTTTTT
  IIIIII      OOOOOO      CCCCCCCC    IIIIII      OOOOOO    PPPPPPPP      OOOOOO      SSSSSSSS  TTTTTTTTTT
    II      OO      OO  CC              II      OO      OO  PP      PP  OO      OO  SS              TT
    II      OO      OO  CC              II      OO      OO  PP      PP  OO      OO  SS              TT
    II      OO      OO  CC              II      OO      OO  PP      PP  OO      OO  SS              TT
    II      OO      OO  CC              II      OO      OO  PP      PP  OO      OO  SS              TT
    II      OO      OO  CC              II      OO      OO  PPPPPPPP    OO      OO    SSSSSS        TT
    II      OO      OO  CC              II      OO      OO  PPPPPPPP    OO      OO    SSSSSS        TT
    II      OO      OO  CC              II      OO      OO  PP          OO      OO          SS      TT
    II      OO      OO  CC              II      OO      OO  PP          OO      OO          SS      TT
    II      OO      OO  CC              II      OO      OO  PP          OO      OO          SS      TT      ....
    II      OO      OO  CC              II      OO      OO  PP          OO      OO          SS      TT      ....
  IIIIII      OOOOOO      CCCCCCCC    IIIIII      OOOOOO    PP            OOOOOO      SSSSSSSS      TT      ....
  IIIIII      OOOOOO      CCCCCCCC    IIIIII      OOOOOO    PP            OOOOOO      SSSSSSSS      TT      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

IOC
V04

Table of contents

IOC
V04-

```
0000          .TITLE  IOCIOPOST - I/O COMPLETION POSTING
0000          .IDENT  'V04-001'
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000
0000   ;++
0000   ; FACILITY: EXECUTIVE, I/O SYSTEM
0000   ;
0000   ; ABSTRACT:
0000   ;       IOCIOPOST IMPLEMENTS THE DEVICE INDEPENDENT COMPLETION PROCESSING FOR
0000   ;       I/O PACKETS.  IT IS INVOKED BY QUEUEING THE PACKET ON THE I/O POST QUEUE
0000   ;       AND TRIGGERING THE IPL$_IOPOST SOFTWARE INTERRUPT.  SOME OF THE IOPOST
0000   ;       OPERATIONS SUCH AS SETTING EVENT FLAGS, UNLOCKING BUFFER PAGES,
0000   ;       RELEASING BUFFERS AND PAGING I/O COMPLETION ARE PERFORMED IN THE IOPOST
0000   ;       INTERRUPT SERVICE ROUTINE, WHILE OTHER OPERATIONS THAT REQUIRE ACCESS
0000   ;       TO PROCESS ADDRESS SPACE ARE PERFORMED BY SENDING A SPECIAL KERNEL AST.
0000   ;
0000   ; ENVIRONMENT:  MODE = KERNEL, RESIDENT
0000   ;
0000   ;--
0000          .SBTTL  HISTORY                 ; DETAILED
0000   ;
0000   ; AUTHOR: R. HUSTVEDT, CREATION DATE: 26-AUG-76
0000   ;
0000   ; MODIFIED BY:
0000   ;
0000   ;       V04-001 SSA0031         Stan Amway              7-Sep-1984
0000   ;               Fix bug introduced by EMD0076 that destroys UCB address
0000   ;               in R0 if encryption key buffer is present.
0000   ;
0000   ;       V03-025 WMC0025         Wayne Cardoza           31-May-1984
0000   ;               Make sure direct I/O completion unlocks at least one page.
0000   ;
0000   ;       V03-024 ACG0422         Andrew C. Goldstein,    1-May-1984  19:35
0000   ;               Fix use of R0 in ACG0421
```

```
0000    ;
0000    ;       V03-023 ACG0421         Andrew C. Goldstein,    20-Apr-1984  14:19
0000    ;               Fix segment byte count limiting in erase QIO's
0000    ;
0000    ;       V03-022 EMD0076         Ellen M. Dusseault      05-Apr-1984
0000    ;               Modify IOPOST to check for a valid status bit for
0000    ;               encryption.  If valid, deallocate nonpaged pool buffer
0000    ;               which contains the encryption key.
0000    ;
0000    ;       V03-021 SSA0021         Stan Amway              22-Mar-1984
0000    ;               Decrement device queue length in UCB.
0000    ;
0000    ;       V03-020 WMC0020         Wayne Cardoza           07-Mar-1984
0000    ;               Move POSTEF to fork context to regain optimization which
0000    ;               avoids reexecution of WAITFR.
0000    ;
0000    ;       V03-019 WMC0019         Wayne Cardoza           28-Dec-1983
0000    ;               Erase QIOs can be physical, logical, or virtual.
0000    ;
0000    ;       V03-018 CDS0003         Christian D. Saether    14-Dec-1983
0000    ;               Add IOC$BUFPOST entry point.  This is used to perform
0000    ;               the iopost level part of i/o posting to be executed as
0000    ;               a subroutine call directly and avoid the iopost software
0000    ;               interrupt entirely.  The F11BXQP is the initial user
0000    ;               of this feature.
0000    ;
0000    ;       V03-017 ROW49597C       Ralph O. Weber          21-SEP-1983
0000    ;               Change PAGEIO_OR_SWAPIO patch (from ROW49597B and ROW49597) to
0000    ;               zero bytes transfered count in the IOSB when status is not
0000    ;               successful and bytes transfered is greater than or equal to
0000    ;               bytes requested.
0000    ;
0000    ;       V03-016 ROW0218         Ralph O. Weber          7-SEP-1983
0000    ;               Change maximum byte count, UCB$L_MAXBCNT, tests to be
0000    ;               unsigned.
0000    ;
0000    ;       V03-015 ADE9005         Alan D. Eldridge        30-May-1983
0000    ;               Changed BSBW to JSB for calls to IOC$MAPVBLK and IOC$CVTLOGPHY.
0000    ;
0000    ;       V03-014 STJ3100         Steven T. Jeffreys,     03-May-1983
0000    ;               -Added local subroutine CHECK_ERASE.
0000    ;               -Do not update IRP$L_SVAPTE for ALL erase I/O's.  This
0000    ;                is an extention of STJ3085.
0000    ;
0000    ;       V03-013 STJ3085         Steven T. Jeffreys,     13-Apr-1983
0000    ;               -Do not update IRP$L_SVAPTE for erase I/O segmented
0000    ;                requests if using the specail erase PPT.
0000    ;               -After segmentation complete, resore original SVAPTE
0000    ;                address to IRP$L_SVAPTE.
0000    ;
0000    ;       V03-012 ROW49597B       Ralph O. Weber          9-APR-1983
0000    ;               Change PAGEIO_OR_SWAPIO from ROW49597 to zero bytes
0000    ;               transfered count when status is not successful and bytes
0000    ;               transfered is greater than or equal to bytes requested.
0000    ;
0000    ;       V03-011 RLRMXBCNTc      Robert L. Rappaport     28-Mar-1983
0000    ;               Verify IRP$L_DIAGBUF is non-zero before assuming that it
```

```
0000     ;              contains the original value of IRP$L_SVAPTE in VIRTUAL_LOGIO.
0000     ;
0000     ;      V03-010 RLRMXBCNTb        Robert L. Rappaport      22-Mar-1983
0000     ;              Check for file oriented device before going to VIRTUAL_LOGIO.
0000     ;
0000     ;      V03-009 RLRMXBCNTa        Robert L. Rappaport      22-Mar-1983
0000     ;              CLRL the byte count in the I/O status before queueing
0000     ;              an IRP back to the ACP in VIRTUAL_LOGIO.
0000     ;
0000     ;      V03-008 RLRMXBCNT         Robert L. Rappaport      11-Mar-1983
0000     ;              Allow for segmentation of Logical I/O (and Virtual)
0000     ;              based on the UCB$L_MAXBCNT field.
0000     ;
0000     ;      V03-007 ROW49597          Ralph O. Weber           26-JAN-1983
0000     ;              Change both VIRTUAL and PAGEIO_OR_SWAPIO to guarantee an error
0000     ;              status in IRP$L_IOST1 whenever the bytes transfered is less
0000     ;              than the bytes requested.  For V3.x, the error will be
0000     ;              SS$_CTRLERR.  After that, it will be SS$_INCSEGTRA.  The check
0000     ;              and error status are required to detect and gracefully
0000     ;              revcover from the instance where a driver returns success
0000     ;              status but bytes transfered is less than bytes requested.
0000     ;              The segmented transfer logic goes berserk when this happens
0000     ;              and eventually crashes the system.
0000     ;
0000     ;      V03-006 STJ3049           Steven T. Jeffreys       06-Jan-1983
0000     ;              Add support for the erase qio.
0000     ;
0000     ;      V03-005 CDS0002           C Saether                12-Oct-1982
0000     ;              Fix bug where R5 was not preserved when queuing
0000     ;              packet to xqp.
0000     ;
0000     ;      V03-004 CDS0001           C Saether                18-Jul-1982
0000     ;              Changes to accomodate XQP mechanism.
0000     ;
0000     ;      V03-003 KDM0002           Kathleen D. Morse        28-Jun-1982
0000     ;              Added $DEVDEF and $SSDEF.
0000     ;
0000     ;      V03-002 LJK45299          Lawrence J. Kenah        2-Jun-1982
0000     ;              Fix deaccess-pending-on-spooled-device logic.
0000     ;
0000     ;
0000     ;**
```

```
         0000          .SBTTL  DECLARATIONS
         0000  ;
         0000  ; INCLUDE FILES:
         0000  ;
         0000          $ACBDEF                                 ; AST CONTROL BLOCK DEFINITIONS
         0000          $AQBDEF                                 ; DEFINE AQB OFFSETS
         0000          $CADEF                                  ; CONDITIONAL ASSEMBLY PARAMETERS
         0000          $CCBDEF                                 ; CCB DEFINITIONS
         0000          $CXBDEF                                 ; DEFINE CXB OFFSETS
         0000          $DCDEF                                  ; DEVICE TYPE CODES
         0000          $DEVDEF                                 ; DEVICE TYPE DEFINITIONS
         0000          $IODEF                                  ; I/O REQUEST CODES
         0000          $IPLDEF                                 ; IPL DEFINITIONS
         0000          $IRPDEF                                 ; IRP DEFINITIONS
         0000          $IRPEDEF                                ; IRPE DEFINITIONS
         0000          $JIBDEF                                 ; JIB DEFINITIONS
         0000          $PCBDEF                                 ; PCB DEFINITIONS
         0000          $PFNDEF                                 ; PFN DATA BASE DEFINITIONS
         0000          $PHDDEF                                 ; PROCESS HEADER DEFINITIONS
         0000          $PRDEF                                  ; PROCESSOR REGISTER DEFINITIONS
         0000          $PRIDEF                                 ; PRIORITY INCREMENT DEFS
         0000          $PTEDEF                                 ; PAGE TABLE ENTRY DEFINITIONS
         0000          $RSNDEF                                 ; DEFINE RESOURCE WAIT NUMBERS
         0000          $SSDEF                                  ; DEFINE SYSTEM STATUS CODES
         0000          $UCBDEF                                 ; DEFINE UCB OFFSETS
         0000          $VADEF                                  ; DEFINE VIRTUAL ADDRESS FIELDS
         0000          $VCBDEF                                 ; DEFINE VCB OFFSETS
         0000          $WCBDEF                                 ; DEFINE WCB OFFSETS
         0000          $WQHDEF                                 ; WAIT QUEUE HEADER DEFINITIONS
         0000
         0000  ;
         0000  ; OWN STORAGE:
         0000  ;
     00000000          .PSECT  $AEXENONPAGED,LONG
         0000  PRITBL:                                         ; TABLE OF PRIORITY INCR CLASSES
   01    0000          .BYTE   PRI$_IOCOM                      ; 0 => DIRECT WRITE
   03    0001          .BYTE   PRI$_TOCOM                      ; 1 => BUFFERED WRITE
   01    0002          .BYTE   PRI$_IOCOM                      ; 2 => DIRECT READ
   04    0003          .BYTE   PRI$_TICOM                      ; 3 => BUFFERED READ
```

```
                        0004            .SBTTL  I/O COMPLETION POSTING
                        0004     ;++
                        0004     ; FUNCTIONAL DESCRIPTION:
                        0004     ;
                        0004     ;       IOC$IOPOST IS INITIATED BY TRIGGERING AN IPL$_IOPOST SOFTWARE
                        0004     ;       INTERRUPT AFTER PLACING A COMPLETED I/O PACKET IN THE IOPOST
                        0004     ;       QUEUE.  IOC$IOPOST PERFORMS ALL APPROPRIATE COMPLETION ACTIVITY
                        0004     ;       REQUIRED FOR THE PACKET EITHER DIRECTLY OR BY QUEUEING KERNEL
                        0004     ;       ASTS TO CONCLUDE PROCESSING IN THE CONTEXT OF THE PROCESS
                        0004     ;       WHEN REQUIRED.
                        0004     ;
                        0004     ; CALLING SEQUENCE:
                        0004     ;
                        0004     ;       SOFTINT #IPL$_IOPOST
                        0004     ;
                        0004     ; INPUT PARAMETERS:
                        0004     ;
                        0004     ;       NONE
                        0004     ;
                        0004     ; IMPLICIT INPUTS:
                        0004     ;
                        0004     ;       IOC$GL_PSFL - IOPOSTING QUEUE
                        0004     ;
                        0004     ; OUTPUT PARAMETERS:
                        0004     ;
                        0004     ;       NONE
                        0004     ;
                        0004     ;--
                        0004
                        0004            .ENABL  LSB
                        0004     IOC$IOPOST::                           ; I/O POSTING INTERRUPT
         7E   54    7D  0004            MOVQ    R4,-(SP)                ; SAVE
         7E   52    7D  0007            MOVQ    R2,-(SP)                ; NORMAL
         7E   50    7D  000A            MOVQ    R0,-(SP)                ; REGISTERS
   55  0000'DF      0F  000D     IOPOST: REMQUE @W^IOC$GL_PSFL,R5       ; GET HEAD OF POST QUEUE
              16    1C  0012            BVC     10$                     ; QUEUE NOT YET EMPTY
         50   8E    7D  0014            MOVQ    (SP)+,R0                ; RESTORE
         52   8E    7D  0017            MOVQ    (SP)+,R2                ; REGISTERS
         54   8E    7D  001A            MOVQ    (SP)+,R4                ; AND EXIT
                    02  001D            REI                             ; IF QUEUE EMPTY
                        001E
            0394    31  001E     5$:    BRW     VIRTUAL_LOGIO           ; PROCESS VIRTUAL (OR LOGICAL) I/O COMPLETIO
                        0021
              61    16  0021     7$:    JSB     (R1)                    ; CALL END ACTION ROUTINE
              E8    11  0023            BRB     IOPOST                  ;
                        0025
           6A A0    B4  0025     8$:    CLRW    UCB$W_QLEN(R0)          ; Device queue length went negative
              18    11  0028            BRB     11$                     ; Reset queue length and continue
                        002A
   51      0C A5    D0  002A     10$:   MOVL    IRP$L_PID(R5),R1        ; GET PID/END ACTION ADDRESS
              F1    19  002E            BLSS    7$                      ; BR IF END ACTION ADDRESS
                        0030                                            ; (SYSTEM SPACE ADDRESSES ARE NEGATIVE)
      51      51    3C  0030            MOVZWL  R1,R1                   ; GET PROCESS INDEX
  54   0000'DF41    D0  0033            MOVL    @W^SCH$GL_PCBVEC[R1],R4 ; AND TRANSLATE TO PCB ADDRESS
   50      1C A5    D0  0039            MOVL    IRP$L_UCB(R5),R0        ; R0 => UCB. (Presets UCB for DIO path)
           6A A0    B7  003D            DECW    UCB$W_QLEN(R0)          ; Decrement device queue length
              E3    19  0040            BLSS    8$                      ; Length went negative, so go adjust
```

```
   0C 2A A5     0F     E1  0042  11$:     BBC     #IRP$V_KEY,IRP$W_STS(R5),12$ ; set, buffer alloc for encryption
                50     DD  0047           PUSHL   R0                          ; Save UCB address
   50    5C A5         D0  0049           MOVL    IRP$L_KEYDESC(R5), R0       ; r0 contains address of alloc buffer
              FFB0'    30  004D           BSBW    EXE$DEANONPAGED             ; deallocate buffer (R0-R3 destroyed)
             50 8ED0       0050           POPL    R0                          ; Restore UCB address
   03 2A A5     00     E1  0053  12$:     BBC     #IRP$V_BUFIO,IRP$W_STS(R5),13$  ; IF CLEAR, DIRECT I/O
              00CA     31  0058           BRW     BUFIO                       ; BUFFERED I/O
             3E A4     B6  005B  13$:     INCW    PCB$W_DIOCNT(R4)            ; UPDATE DIRECT I/O COUNT
   53        2C A5     D0  005E           MOVL    IRP$L_SVAPTE(R5),R3         ; GET ADDRESS OF FIRST PTE
                           0062
                           0062           ASSUME  IRP$V_PAGIO LE 7
                           0062           ASSUME  IRP$V_SWAPIO LE 7
      2A A5   44 8F    93  0062           BITB    #<IRP$M_PAGIO ! IRP$M_SWAPIO>,IRP$W_STS(R5) ; PAGIO OR SWAPIO?
                61     12  0067           BNEQ    PAGIO_OR_SWAPIO
                           0069
                           0069  ;
                           0069  ; DIRECT I/O COMPLETION
                           0069  ;
                           0069
                53     D5  0069  DIRIO:   TSTL    R3                          ; PTE ADDRESS VALID?
                46     13  006B           BEQL    18$                         ; IF EQL NO PAGES TO UNLOCK
   51        32 A5     D0  006D           MOVL    IRP$L_BCNT(R5),R1           ; GET REQUESTED TRANSFER BYTE COUNT
   52        30 A5     3C  0071           MOVZWL  IRP$W_BOFF(R5),R2           ; GET BYTE OFFSET IN PAGE
                08     E0  0075           BBS     #IRP$V_PHYSIO,-
         1C 2A A5          0077                   IRP$W_STS(R5),UNLOCK        ; BRANCH IF PHYSICAL I/O
                0E     E1  007A           BBC     #DEV$V_FOD,-                ; If NOT file oriented, go to UNLOCK.
         17 38 A0          007C                   UCB$L_DEVCHAR(R0),UNLOCK;   (R0 preloaded in common DIO/BIO path)
         9B 38 A5      E9  007F           BLBC    IRP$L_IOST1(R5),5$          ; BRANCH IF ERROR IN VIRT. OR LOG. REQUEST
            46 A5      B5  0083           TSTW    IRP$L_OBCNT+2(R5)           ; WAS ORIGINAL COUNT > 64K?
                07     13  0086           BEQL    14$                         ; EQL IMPLIES NO
            3A A5      D1  0088           CMPL    IRP$L_IOST1+2(R5),-         ; LONGWORD COMPARE FOR > 64K OBCNT
            44 A5          008B                   IRP$L_OBCNT(R5)             ; IF COMPLETED ORIGINAL BYTE COUNT
                           008D                                               ; THEN NO SPECIAL VIRTUAL PROCESSING
                05     11  008D           BRB     16$                         ; BRANCH AROUND TO COMMON 'BNEQ'
                           008F  14$:
            3A A5      B1  008F           CMPW    IRP$L_IOST1+2(R5),-         ; *NOTE 'CMPW' DUE TO CODE PATH FOR <64K BCN
            44 A5          0092                   IRP$L_OBCNT(R5)             ; IF COMPLETED ORIGINAL BYTE COUNT
                           0094                                               ; THEN NO SPECIAL VIRTUAL PROCESSING
                           0094  16$:
                88     12  0094           BNEQ    5$                          ; OTHERWISE DO THE SEGMENTED COMPLETION
   51     01FF C142    9E  0096  UNLOCK:  MOVAB   511(R1)[R2],R1              ; COMBINE OFFSET AND COUNT AND ROUND
   51     51    F7 8F  78  009C           ASHL    #-VA$S_BYTE,R1,R1           ; CONVERT TO NUMBER OF PAGES
                02     12  00A1           BNEQ    165$                        ; CHECK FOR AT LEAST ONE PAGE
                51     D6  00A3           INCL    R1                          ; THE FDT ROUTINE LOCKED ONE PAGE
                0A     E1  00A5  165$:    BBC     #IO$V_ERASE,-               ; BRANCH IF DEFINITELY NOT AN ERASE
         06 20 A5          00A7                   IRP$W_FUNC(R5),17$          ;
              066C     30  00AA           BSBW    CHECK_ERASE                 ; IS THIS AN ERASE FUNCTION?
             0B 50     E8  00AD           BLBS    R0,19$                      ; BRANCH IF IT IS AN ERASE
              FF4D'    30  00B0  17$:     BSBW    MMG$UNLOCK                  ; UNLOCK PAGES
                0B     E1  00B3  18$:     BBC     #IRP$V_EXTEND,-
         03 2A A5          00B5                   IRP$W_STS(R5),19$           ; BRANCH IF NO IRPE'S ATTACHED
              06B3     30  00B8           BSBW    UNLOCK_MORE                 ; UNLOCK AREAS DESCRIBED IN IRPE'S
                           00BB  19$:                                         ; REFERENCE LABEL
                           00BB
                           00BB           .IF DF  CA$_MEASURE_IOT
                           00BB
              FF42'    30  00BB           BSBW    PMS$END_RQ                  ; INSERT END OF I/O REQUEST MESSAGE
```

```
                          00BE
                          00BE         .ENDC
                          00BE
      18 A5   0651'CF  9E 00BE         MOVAB   W^DIRPOST,ACB$L_KAST(R5) ; SET SPECIAL KERNEL AST ADDRESS
                 009A  31 00C4         BRW     40$                      ;
                          00C7
                          00C7 BRW_QNXTSEG:
                 034A  31 00C7         BRW     QNXTSEG                  ; GO DO THE NEXT VIRTUAL SEGMENT
                          00CA
                          00CA ;
                          00CA ;       PAGE I/O OR SWAP I/O COMPLETION
                          00CA ;
                          00CA
                          00CA PAGIO_OR_SWAPIO:                         ; HERE WE ASSUME DISK I/O FOR PAGING
                          00CA                                          ;   AND SWAPPING AND WE FURTHER RELY
                          00CA                                          ;   ON THE FACT THAT ALL DISK DRIVERS
                          00CA                                          ;   TRADITIONALLY RETURN ZERO IN THE 2ND
                          00CA                                          ;   LONGWORD OF THE I/O STATUS BLOCK FOR
                          00CA                                          ;   DATA TRANSFER OPERATIONS.  THEREFORE
                          00CA                                          ;   THIS IS COMPATIBLE WITH DISK CLASS
                          00CA                                          ;   DRIVER WHICH GROWS THE # OF BYTES
                          00CA                                          ;   TRANSFERRED FIELD IN THE IOSB TO A
                          00CA                                          ;   FULL LONGWORD.
         51   3A A5    D0 00CA         MOVL    IRP$L_IOST1+2(R5), R1    ; Get bytes transfered.
          12  38 A5    E9 00CE         BLBC    IRP$L_IOST1(R5), 21$     ; Branch if transfer not successful.
         44 A5    51   D1 00D2         CMPL    R1, IRP$L_OBCNT(R5)      ; If completed whole transfer, skip
                  3E   13 00D6         BEQL    26$                      ; all this segmenting junk.
         32 A5    51   D1 00D8         CMPL    R1, IRP$L_BCNT(R5)       ; Bytes transfered = bytes requested?
                  11   13 00DC         BEQL    23$                      ; Branch if equal.
      38 A5    2234 8F B0 00DE         MOVW    #SS$_INCSEGTRA, -        ; Else, change success status
                          00E4                 IRP$L_IOST1(R5)          ; to error status.
                          00E4                                          ; For the error cases:
         32 A5    51   D1 00E4 21$:    CMPL    R1, IRP$L_BCNT(R5)       ; Bytes transfered < bytes requested?
                  05   1F 00E8         BLSSU   23$                      ; Branch if less than.
                  51   D4 00EA         CLRL    R1                       ; Else, assume no bytes transfered.
               3A A5   D4 00EC         CLRL    IRP$L_IOST1+2(R5)        ; Clear bytes transfered in IRP too.
         40 A5    51   C0 00EF 23$:    ADDL    R1, IRP$L_ABCNT(R5)      ; Update accumulated byte count.
   51    51    17  09  EF 00F3         EXTZV   #VA$V_VPN, -             ; Convert bytes transfered to
                          00F8                 #<32-VA$V_VPN>, R1, R1   ; pages transfered.
         48 A5    51   C0 00F8         ADDL    R1,IRP$L_SEGVBN(R5)      ; NEXT STARTING VBN (OR ERROR VBN)
          09  38 A5    E9 00FC         BLBC    IRP$L_IOST1(R5),24$      ; BRANCH IF ERROR
              40 A5    C3 0100         SUBL3   IRP$L_ABCNT(R5),-
              44 A5       0103                 IRP$L_OBCNT(R5),-        ; CALCULATE REMAINING BYTE
              32 A5       0105                 IRP$L_BCNT(R5)           ; COUNT TO BE TRANSFERRED
                 BE    12 0107         BNEQ    BRW_QNXTSEG              ; BRANCH IF ANOTHER SEGMENT TO DO
                          0109 ;
                          0109 ; LAST SEGMENT COMPLETED OR ERROR
                          0109 ;
                          0109 24$:
              40 A5    D0 0109         MOVL    IRP$L_ABCNT(R5),-
              3A A5       010C                 IRP$L_IOST1+2(R5)        ; SET BYTES TRANSFERRED
         53   4C A5    D0 010E         MOVL    IRP$L_DIAGBUF(R5),R3     ; GET SAVED SVAPTE
      2C A5       53   D0 0112         MOVL    R3,IRP$L_SVAPTE(R5)      ; AND PUT IT BACK
                          0116 26$:
                          0116         .IF DF  CA$_MEASURE_IOT
                          0116
                FEE7'  30 0116         BSBW    PMS$END_RQ               ; INSERT END OF I/O REQUEST MESSAGE
```

```
                         0119
                         0119          .ENDC
                         0119
76 2A A5     02      E0  0119          BBS     #IRP$V_PAGIO,IRP$W_STS(R5),PAGIO ; BRANCH IF PAGE I/O
                         011E
                         011E   ;
                         011E   ;      SWAP I/O COMPLETION
                         011E   ;
                         011E
18 A5     14 A5      D0  011E          MOVL    IRP$L_ASTPRM(R5),ACB$L_KAST(R5) ; SET KERNEL AST ADDRESS
             3C      11  0123          BRB     40$                             ; AND ENQUEUE AST
                         0125
                         0125   ;
                         0125   ; BUFFERED I/O COMPLETION
                         0125   ;
                         0125
   00000161'EF       9F  0125   BUFIO: PUSHAB  40$                             ; 'INLINE' SUBROUTINE CALL.
                         012B
                         012B   ;
                         012B   ; THE FOLLOWING PIECE OF CODE MAY BE CALLED AS A SUBROUTINE DIRECTLY
                         012B   ; TO DO THE PART OF BUFFERED I/O COMPLETION THAT NORMALLY EXECUTES
                         012B   ; AS A RESULT OF AN IOPOST SOFTWARE INTERRUPT.
                         012B   ;
                         012B   ; THE F11BXQP, FOR EXAMPLE, EXECUTES VIRTUAL FILE SYSTEM FUNCTIONS
                         012B   ; IN PROCESS CONTEXT.  THERE IS NO NEED FOR THE IOPOST INTERRUPT
                         012B   ; AND SPECIAL KERNEL AST TO POST I/O COMPLETION.  AFTER RETURNING
                         012B   ; FROM THIS SUBROUTINE, THE F11BXQP WILL DO A
                         012B   ;
                         012B   ;      JSB     @ACB$L_KAST (R5)
                         012B   ;
                         012B   ; TO COMPLETE POSTING THE I/O COMPLETION.
                         012B   ; BOTH THE IOPOST SOFTWARE INTERRUPT AND THE SPECIAL KERNEL COMPLETION
                         012B   ; AST ARE AVOIDED.
                         012B   ;
                         012B   ; THE CALLER SHOULD TEST IRP$L_PID AND POST A NORMAL IOPOST INTERRUPT
                         012B   ; IF IT IS NEGATIVE, AS THAT CASE IS NOT HANDLED HERE.
                         012B   ;
                         012B   ; THE F11BXQP CODE THAT USES THIS ROUTINE IS IN [F11X.SRC]IODONE.MAR.
                         012B   ;
                         012B   ; IPL = IPL$_ASTDEL TO BLOCK PROCESS DELETION (PREVENT LOSS OF IRP).
                         012B   ; R4 = PCB ADDRESS
                         012B   ; R5 = IRP ADDRESS
                         012B   ;
                         012B
                         012B   IOC$BUFPOST::
          3A A4      B6  012B          INCW    PCB$W_BIOCNT(R4)                ; UPDATE BUFFERED I/O COUNT
03 2A A5     0C      E1  012E          BBC     #IRP$V_FILACP,IRP$W_STS(R5),NOTACP ; BR IF NOT ACP I/O
          3E A4      B6  0133          INCW    PCB$W_DIOCNT(R4)                ; RESTORE DIRECT I/O COUNT
                         0136   NOTACP:                                        ;
                         0136
                         0136          .IF DF  CA$_MEASURE_IOT
                         0136
          FEC7'      30  0136          BSBW    PMS$END_RQ                      ; INSERT END OF I/O REQUEST MESSAGE
                         0139
                         0139          .ENDC
                         0139
50     0080 C4       D0  0139          MOVL    PCB$L_JIB(R4),R0                ; GET JIB ADDRESS
```

```
         51    30 A5  3C 013E  426      MOVZWL  IRP$W_BOFF(R5),R1          ; Convert I/O byte count to a longword.
         20 A0    51  C0 0142  427      ADDL    R1,JIB$L_BYTCNT(R0)        ; Update Byte Count Quota.
         50    2C A5  D0 0146  428      MOVL    IRP$L_SVAPTE(R5),R0        ; ANY BUFFER SPECIFIED?
                  0E  13 014A  429      BEQL    30$                        ; IF EQL NO
   18 A5    056F'CF   9E 014C  430      MOVAB   W^BUFPOST,ACB$L_KAST(R5)   ; ASSUME READ FUNCTION
   09 2A A5     01    E0 0152  431      BBS     #IRP$V_FUNC,IRP$W_STS(R5),35$  ; IF SET, READ FUNCTION
              FEA6'   30 0157  432      BSBW    EXE$DEANONPAGED            ; DEALLOCATE WRITE BUFFER
   18 A5    0651'CF   9E 015A  433 30$: MOVAB   W^DIRPOST,ACB$L_KAST(R5)   ; SET SPECIAL KERNEL AST ADDRESS
                      05 0160  434 35$: RSB                                ; RETURN TO PROCESS CONTEXT IOPOSTING
                         0161  435                                         ; PROCESS, OR CONTINUE INLINE IF THIS
                         0161  436                                         ; IS NORMAL IOPOST SOFTWARE INTERRUPT.
50   2A A5    02     00  EF 0161  437 40$: EXTZV   #IRP$V_BUFIO,#2,IRP$W_STS(R5),R0  ; GET PACKET TYPE
   03 2A A5    09    E0 0167  438      BBS     #IRP$V_TERMIO,IRP$W_STS(R5),50$ ; BR IF TERMINAL I/O
         50    01     AA 016C  439      BICW    #1,R0                      ; ELSE TREAT AS NORMAL I/O COMPLETION
                         016F  440 50$:                                    ; FOR PRIORITY INCREMENT SELECTION
         51    0C A5  D0 016F  441      MOVL    IRP$L_PID(R5),R1           ; PROCESS IDENTIFICATION
    52   FE88 CF40    9A 0173  442      MOVZBL  PRITB[R0],R2               ; SET PRIORITY INCREMENT CLASS
         53    22 A5  9A 0179  443      MOVZBL  IRP$B_EFN(R5),R3           ; GET EVENT FLAG NUMBER
                         017D  444      DSBINT  #IPL$_SYNCH                ; PREVENT INTERRUPT FROM MP SECONDARY
              FE7A'   30 0183  445      BSBW    SCH$POSTEF                 ; AND POST IT
      0B A5    80 8F  88 0186  446      BISB    #^X80,ACB$B_RMOD(R5)       ; SET INTERNAL AST FLAG
              FE72'   30 018B  447      BSBW    SCH$QAST                   ; NOW QUEUE THE KERNEL AST
                         018E  448      ENBINT
               FE79   31 0191  449      BRW     IOPOST                     ; GET NEXT PACKET TO POST
                         0194  450      .DSABL  LSB
```

```
                       0194          .SBTTL  PAGIO - PAGE I/O COMPLETION
                       0194  ;
                       0194  ; PAGING I/O COMPLETION
                       0194  ;
                       0194  ; INPUTS:
                       0194  ;
                       0194  ;       R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
                       0194  ;       R4 = PROCESS CONTROL BLOCK ADDRESS
                       0194  ;       R5 = I/O REQUEST PACKET ADDRESS
                       0194  ;
                       0194  ;               FOR PAGE READ COMPLETION, THE FOLLOWING LOCATIONS IN THE
                       0194  ;       I/O REQUEST PACKET HAVE SPECIAL SIGNIFICANCE.
                       0194  ;
                       0194  ;       IRP$L_ASTPRM    = ORIGINAL PROCESS PAGE TABLE ENTRY BACKING STORE
                       0194  ;                         ADDRESS IF PAGE WAS A COPY ON REFERENCE PAGE.
                       0194  ;                         PFN$V_GBLBAK SET IF IT WAS GLOBAL CRF
                       0194  ;                       = 0 IF NOT A COPY ON REFERENCE PAGE
                       0194  ;       IRP$L_AST       = MASTER PTE CONTENTS IF GLOBAL CRF (>0)
                       0194  ;                       = SLAVE PTE ADDRESS IF GLOBAL NOT CRF (<0)
                       0194  ;                       = 0 IF NOT GLOBAL
                       0194  ;
                       0194  ;               FOR PAGE WRITE COMPLETION, THE FOLLOWING LOCATIONS IN
                       0194  ;       THE I/O REQUEST PACKET HAVE SIGNIFICANCE.
                       0194  ;
                       0194  ;       IRP$B_RMOD      = REQUEST MODE ! ACB$V_QUOTA.  IF ACB$V_QUOTA IS SET,
                       0194  ;                         PROCESS REQUESTED AN AST ON PAGE WRITE COMPLETION
                       0194  ;       IRP$L_AST       = AST ADDRESS IF REQUESTED
                       0194  ;       IRP$L_ASTPRM    = AST PARAMETER IF SPECIFIED
                       0194  ;       IRP$L_IOSB      = ADDRESS OF I/O STATUS BLOCK IF SPECIFIED.  IF
                       0194  ;                         NON-ZERO, THEN PROCESS EXPECTS I/O STATUS RETURNED.
                       0194  ;
          7E   56   7D 0194  PAGIO:  MOVQ    R6,-(SP)                        ; SAVE SOME MORE REGISTERS
          56   55   D0 0197          MOVL    R5,R6                           ; USE R6 FOR IRP ADDRESS
                       019A
                       019A          SETIPL  #IPL$_SYNCH                     ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
       55   6C A4   D0 019D          MOVL    PCB$L_PHD(R4),R5                ; USE R5 FOR PROCESS HEADER ADR
               09   EF 01A1          EXTZV   #VA$V_VPN,-
               17      01A3                  #<32-VA$V_VPN>,-                ; FORM PAGE COUNT
       57   3A A6      01A4                  IRP$L_IOST1+2(R6),R7            ; OF THE DATA TRANSFERRED
    33 2A A6   01   E0 01A7          BBS     #IRP$V_FUNC,IRP$W_STS(R6),PAGRD_DONE ; BRANCH IF PAGE READ
                       01AC  ;
                       01AC  ; PAGE WRITE COMPLETE - R7 = NUMBER OF PAGES
                       01AC  ;       CONDITION CODES SET FROM LOAD OF R7
                       01AC  ;
               1F   13 01AC          BEQL    60$                             ; BRANCH IF NO PAGES SUCCESSFULLY TRANSFERRE
    50   63   15   00   EF 01AE      EXTZV   #PTE$V_PFN,#PTE$S_PFN,(R3),R0 ;GET PFN FROM PTE
    00000000'EF   50   D1 01B3       CMPL    R0,MMG$GL_MAXPFN                ;IS THIS PAGE IN SHARED MEMORY?
               11   1A 01BA          BGTRU   60$                             ;BR IF PAGE IN SH MEM, NO PFN DATABASE
               53   DD 01BC  20$:    PUSHL   R3                              ; SAVE SVAPTE
             01C3   30 01BE          BSBW    PFN_IO_DONE                     ; SET PFN DATA BASE
                       01C1  ;
                       01C1  ; CONDITION CODES SET FROM DECREF
                       01C1  ;
               03   14 01C1          BGTR    40$                             ; BRANCH IF REFCNT NOT 0
            FE3A'   30 01C3          BSBW    MMG$RELPFN                      ; RELEASE THE PAGE
       53   8E   04   C1 01C6  40$:  ADDL3   #4,(SP)+,R3                     ; GET NEXT PTE ADDRESS
            EF 57   F5 01CA          SOBGTR  R7,20$                          ; DO THE NEXT PAGE IF ANY
```

```
57   08 A6   00C4 8F   A3 01CD  60$:   SUBW3  #IRP$C_LENGTH,IRP$W_SIZE(R6),R7 ; IF EXTENDED I/O PACKET
                          01D4                                                ; THEN COMPLETION IS DONE BY
                          01D4                                                ; SPECIAL UPDATE SECTION KERNEL AST
          04 38 A6     E9 01D4         BLBC   IRP$L_IOST1(R6),PAGWRT_ERR ; BRANCH IF PAGE WRITE ERROR
                          01D8  ;
                          01D8  ; CONDITION CODES SET FROM LOAD OF R7
                          01D8  ;
                          01D8  PAGWRT_ERR_DONE:
                   68  13 01D8         BEQL   PAGIO_DONE1           ; BRANCH IF NOT, COMPLETE THE I/O HERE
                   6D  11 01DA         BRB    PAGIO_DONE2           ; COMPLETE I/O IN PROCESS CONTEXT
                          01DC  PAGWRT_ERR:
                 0150  31 01DC         BRW    PAGWRT_ERR1
                          01DF  ;
                          01DF  ; PAGE READ COMPLETE - R7 = NUMBER OF PAGES
                          01DF  ;        CONDITION CODES SET FROM LOAD OF R7
                          01DF  ;
                          01DF  PAGRD_DONE:
                   3C  13 01DF         BEQL   100$                  ; BRANCH IF NO PAGES SUCCESSFULLY TRANSFERRE
                 01A0  30 01E1  20$:   BSBW   PFN_IO_DONE           ; RECORD PAGE READ DONE
                          01E4  ;
                          01E4  ; CONDITION CODES SET FROM DECREF
                          01E4  ;
                   11  14 01E4         BGTR   30$                   ; BRANCH IF REFCNT NOT ZERO
                          01E6  ;
                          01E6  ; NO MORE REFERENCES FOR THIS PAGE, DON'T MAKE IT VALID, RELEASE IT
                          01E6  ;
                04 A3  DF 01E6         PUSHAL 4(R3)                 ; SAVE PTE ADR FOR NEXT PTE
                FE14'  30 01E9         BSBW   MMG$RELPFN            ; RELEASE THE PFN
                   08  BA 01EC         POPR   #^M<R3>               ; RECOVER PTE FOR NEXT PAGE IN CLUSTER
51      10 A6      04  C1 01EE         ADDL3  #4,IRP$L_AST(R6),R1   ; GLOBAL PAGE?
                   25  18 01F3         BGEQ   80$                   ; BRANCH IF IT ISN'T
                   1F  11 01F5         BRB    60$                   ; YES, SET CONTEXT FOR NEXT PAGE IN CLUSTER
      0000'DF40    07  88 01F7  30$:   BISB   #PFN$C_ACTIVE,@W^PFN$AB_STATE[R0] ; PAGE IS NOW ACTIVE
         00 50     1F  E2 01FD         BBSS   #PTE$V_VALID,R0,40$   ; TURN VALID ON WITH PFN
            83     50  C8 0201  40$:   BISL   R0,(R3)+              ; SET VALID IN PTE
                          0204                                      ; NEXT PTE ADDRESS IN R3
51      10 A6          D0 0204         MOVL   IRP$L_AST(R6),R1      ; GLOBAL PAGE?
                   10  18 0208         BGEQ   80$                   ; BRANCH IF NOT
                          020A  ;
                          020A  ; PAGE IS A GLOBAL PAGE, R1 = PROCESS PTE, MUST MAKE IT VALID TOO
                          020A  ;
52   61   867FFFFF 8F  CB 020A         BICL3  #^C<PTE$M_PROT ! PTE$M_OWN>,(R1),R2 ; PROTECTION AND OWNER FIELDS
        81    52   50  C9 0212         BISL3  R0,R2,(R1)+           ; MAKE PROCESS PTE VALID
         10 A6     51  D0 0216  60$:   MOVL   R1,IRP$L_AST(R6)      ; SET UP FOR NEXT PAGE IN CLUSTER
            C4     57  F5 021A  80$:   SOBGTR R7,20$                ; DO THE NEXT PAGE IF ANY
         7F 38 A6      E9 021D  100$:  BLBC   IRP$L_IOST1(R6),PAGRD_ERR ; BRANCH IF PAGE READ ERROR
                          0221  ;
                          0221  ; LAST PAGE IN CLUSTER HAS BEEN PROCESSED, COMPLETE THE PROCESSING
                          0221  ; ASSOCIATED WITH THE TRANSFER AS A WHOLE.
                          0221  ;
                          0221  PAGIO_DONE:
51      14 A6          D0 0221         MOVL   IRP$L_ASTPRM(R6),R1   ; COPY ON REFERENCE SECTION?
                   1B  13 0225         BEQL   20$                   ; BRANCH IF NOT
         0B 51     17  E1 0227         BBC    #PFN$V_GBLBAK,R1,10$  ; BRANCH IF NOT GBL CRF
55      00000000'FF    DE 022B         MOVAL  @MMG$GL_SYSPHD,R5     ; SYSTEM HDR FOR GBL CRF PAGE
51      10 A6          D0 0232         MOVL   IRP$L_AST(R6),R1      ; CONTENTS OF GBL PTE FOR GBL CRF
      51     51        32 0236  10$:   CVTWL  R1,R1                 ; SECTION INDEX
```

```
          09   EF  0239         EXTZV   #VA$V_VPN,-
          17       023B                 #<32-VA$V_VPN>,-            ; PAGE COUNT FROM
     50   3A A6    023C                 IRP$L_IOST1+2(R6),R0        ; BYTE COUNT TRANSFERRED
       FDBE'   30  023F         BSBW    MMG$SUBSECREF               ; SUBTRACT R0 FROM SECTION REFERENC COUNT
                   0242 ;
                   0242 ; REPORT THAT PAGE I/O HAS COMPLETED.
                   0242 ;
                   0242 ; NORMALLY IT IS ONLY NECESSARY TO REPORT "PAGE FAULT COMPLETE"
                   0242 ; TO THE PROCESS THAT INITIATED THE I/O, BUT FOR SYSTEM PAGES
                   0242 ; AND FOR GLOBAL PAGES, MULTIPLE FAULTS CAN OCCUR FOR THE SAME
                   0242 ; PAGE WHILE IT IS ON ITS WAY INTO MEMORY.  ALL PROCESSES WHICH
                   0242 ; FAULT THE PAGE WHILE ITS STATE IS "READ IN PROGRESS" GET QUEUED
                   0242 ; ON THE COLLISION PAGE QUEUE, AND THE COLLISION BIT IS SET IN THE
                   0242 ; TYPE BYTE OF THE PFN DATA BASE.  THIS ROUTINE ALSO REPORTS THE
                   0242 ; COLLISION PAGE AVAILABLE EVENT TO ALL PROCESSES ON THE COLLISION
                   0242 ; QUEUE, IF THE COLLISION BIT IS SET.
                   0242 ;
                   0242 20$:
                   0242 PAGIO_DONE1:
     52   00   9A  0242         MOVZBL  #PRI$_NULL,R2               ; SET FOR NULL PRIORITY INCREMENT
                   0245         RPTEVT  PFCOM                       ; REPORT PAGE FAULT COMPLETE
                   0249 ;
                   0249 ; IRP$W_BOFF WAS INCREMENTED IF ANY OF THE PAGES HAD THE COLLISION BIT SET
                   0249 ;
                   0249 ;       R7 = NON ZERO IF SUPPOSED TO ISSUE KERNEL AST
                   0249 ;            USED ONLY FOR PAGE WRITE COMPLETION
                   0249 ;            BUT MUST BE ZERO FOR PAGE READ COMPLETION
                   0249 ;
                   0249 PAGIO_DONE2:
       30 A6   B5  0249         TSTW    IRP$W_BOFF(R6)              ; ANY PAGES WITH COLLISION BIT SET?
          15   13  024C         BEQL    60$                         ; BRANCH IF NOT
          54   DD  024E         PUSHL   R4                          ; SAVE PCB ADDRESS
    0008'CF    B5  0250 40$:    TSTW    W^SCH$GQ_COLPGWQ+WQH$W_WQCNT ; ANYONE WAITING?
          0B   15  0254         BLEQ    50$                         ; BRANCH IF NOT
 54 0000'CF    D0  0256         MOVL    W^SCH$GQ_COLPGWQ,R4         ; GET NEXT PCB
                   025B         RPTEVT  COLPGA                      ; REPORT "COLLISION PAGE AVAILABLE"
          EF   11  025F         BRB     40$                         ; REPEAT UNTIL QUEUE IS EMPTY
          10   BA  0261 50$:    POPR    #^M<R4>                     ; RESTORE SAVED PCB ADDRESS
                   0263 60$:    SETIPL  #IPL$_IOPOST                ; LOWER TO I/O POST LEVEL
          57   D5  0266         TSTL    R7                          ; EXHAUSTED PAGE COUNT NON-ZERO?
          16   12  0268         BNEQ    PAGIO_KAST                  ; BRANCH IF YES, COMPLETE I/O IN PROCESS
    7E 38 A6   E9  026A         BLBC    IRP$L_IOST1(R6),PAGIO_ERR   ; BRANCH IF MORE ERROR PROCESSING TO DO
       50 56   D0  026E         MOVL    R6,R0                       ; GET PACKET ADDRESS FOR RELEASE
       56 8E   7D  0271         MOVQ    (SP)+,R6                    ; RESTORE SAVED REGISTERS
                   0274 ;
                   0274 ; R0 = I/O REQUEST PACKET ADDRESS
                   0274 ;
                   0274 PAGIO_ERR_DONE:
       FD89'   30  0274         BSBW    EXE$DEANONPAGED             ; AND RELEASE IT
     50   01   3C  0277         MOVZWL  #RSN$_ASTWAIT,R0            ; SET AST WAIT RESOURCE WAIT NUMBER
       FD83'   30  027A         BSBW    SCH$RAVAIL                  ; SET RESOURCE AVAILABLE
        FD8D   31  027D         BRW     IOPOST                      ; CONTINUE TO PROCESS POST QUEUE
                   0280 ;
                   0280 ; COMPLETE THE PAGE WRITE IN THE PROCESS CONTEXT
                   0280 ;
                   0280 PAGIO_KAST:
     55   56   D0  0280         MOVL    R6,R5                       ; I/O PACKET ADDRESS BACK TO NORMAL REG
```

```
          56     8E  7D 0283          MOVQ    (SP)+,R6                     ; RESTORE SAVED REGISTERS
          51  0C A5  D0 0286          MOVL    IRP$L_PID(R5),R1             ; PROCESS ID FOR ISSUING KERNEL AST
   18 A5   00000000'EF  9E 028A       MOVAB   MMG$UPDSECAST,ACB$L_KAST(R5) ; ADDRESS TO START KERNEL AST
          52     01  9A 0292          MOVZBL  #PRI$_IOCOM,R2               ; PRIORITY INCREMENT
       0B A5  80 8F  88 0295          BISB    #^X80,ACB$B_RMOD(R5)         ; SET INTERNAL AST FLAG
               FD63'  30 029A         BSBW    SCH$QAST                     ; NOW QUEUE THE KERNEL AST
               FD6D  31 029D          BRW     IOPOST                       ; GET NEXT PACKET TO POST
                        02A0  ;
                        02A0  ; PAGE READ ERROR - CLEAN UP LOGIC
                        02A0  ;
                        02A0  ;       R3 = PTE ADDRESS OF BAD PAGE
                        02A0  ;       R4 = PCB ADDRESS
                        02A0  ;       R5 = PROCESS HEADER ADDRESS
                        02A0  ;       R6 = I/O REQUEST PACKET ADDRESS
                        02A0  ;       R7 = 0 AND MUST BE PRESERVED
                        02A0  ;       IRP$L_AST(R6) = PROCESS PTE ADR OF BAD PAGE IF GLOBAL PAGE
                        02A0  ;       IRP$L_ASTPRM(R6) = GPTX FOR START OF TRANSFER IF GLOBAL CRF
                        02A0  ;
                        02A0  PAGRD_ERR:
               00E1  30 02A0          BSBW    PFN_IO_DONE                  ; COMPLETE THE I/O FOR ERR PAGE
                 14  90 02A3          MOVB    #<PFN$M_DELCON ! PFN$C_RDERR>,- ; SET PAGE TO
          0000'DF40     02A5                  @W^PFN$AB_STATE[R0]          ; READ ERROR STATE
          51  14 A6  D0 02A9          MOVL    IRP$L_ASTPRM(R6),R1          ; GET BACKING STORE ADR IF CRF
                 18  13 02AD          BEQL    120$                         ; BRANCH IF NOT COPY ON REFERENCE
       0E 51     17  E1 02AF          BBC     #PFN$V_GBLBAK,R1,100$        ; BRANCH IF NOT GLOBAL CRF
                 09  EF 02B3          EXTZV   #VA$V_VPN,-
                 17     02B5                  #<32-VA$V_VPN>,-             ; ADJUST GPTX BY
          52  3A A6     02B6                  IRP$L_IOST1+2(R6),R2         ; TRANSFERRED PAGE COUNT
          51     52  C0 02B9          ADDL    R2,R1                        ; TO GET CORRECT GPTX FOR BAD PAGE
   14 A6  51     01  C1 02BC          ADDL3   #1,R1,IRP$L_ASTPRM(R6)       ; SET GPTX FOR START OF NEXT TRANSFER
   0000'DF40     51  D0 02C1  100$:   MOVL    R1,@W^PFN$AL_BAK[R0]         ; FIX BACKING STORE ADDRESS
              10 A6  D5 02C7  120$:   TSTL    IRP$L_AST(R6)                ; IF GLOBAL PAGE (NOT CRF)
                 04  18 02CA          BGEQ    140$
       10 A6     04  C0 02CC          ADDL    #4,IRP$L_AST(R6)             ; THEN SKIP OVER PROCESS PTE ADR
          0000'DF40  B5 02D0  140$:   TSTW    @W^PFN$AW_REFCNT[R0]         ; IS THIS THE LAST REFERENCE?
                 12  14 02D5          BGTR    160$                         ; BRANCH IF NOT
          0000'DF40  B5 02D7          TSTW    @W^PFN$AW_SWPVBN[R0]         ; IF THIS PROCESS HAS BEEN SWAPPED OUT
                 08  13 02DC          BEQL    150$
          52     02  9A 02DE          MOVZBL  #PFN$C_BADPAGLST,R2          ; THEN PUT THIS PAGE IN LIMBO
              FD1C'  30 02E1          BSBW    MMG$INSPFNT                  ; ON THE BAD PAGE LIST
                 03  11 02E4          BRB     160$
              FD17'  30 02E6  150$:   BSBW    MMG$RELPFN                   ; OTHERWISE RELEASE THE PAGE
               FF35  31 02E9  160$:   BRW     PAGIO_DONE                   ; COMPLETE THIS PORTION OF THE PAGE READ
                        02EC  ;
                        02EC  ; DO THE REMAINING SEGMENT OF THE I/O FOR A PAGE READ OR WRITE ERROR
                        02EC  ; SKIP OVER THE PORTION THAT WAS TRANSFERRED SUCCESSFULLY AND SKIP OVER
                        02EC  ; THE PAGE IN ERROR WHICH WAS DEALT WITH BY EITHER PAGRD_ERR OR
                        02EC  ; PAGWRT_ERR AND SET UP TO TRANSFER THE REMAINING PAGES IF ANY.
                        02EC  ; NOTE THAT FOR PAGE WRITE ERRORS THE REST OF THE TRANSFER IS NOT DONE
                        02EC  ; IF I/O COMPLETION STATUS IS RETURNED TO THE PROCESS.
                        02EC  ;
                        02EC  PAGIO_ERR:
          55     56  D0 02EC          MOVL    R6,R5                        ; IRP ADDRESS
          56     8E  7D 02EF          MOVQ    (SP)+,R6                     ; RESTORE ADDITIONAL SAVED REGISTERS
                 09  EF 02F2          EXTZV   #VA$V_VPN,-
                 17     02F4                  #<32-VA$V_VPN>,-             ; GET PAGE COUNT TRANSFERRED
          51  3A A5     02F5                  IRP$L_IOST1+2(R5),R1
```

```
             51   D6  02F8          INCL    R1                            ; COUNT THE ERROR PAGE AS DONE
     50  51   09  9C  02FA          ROTL    #9,R1,R0                      ; BYTE COUNT COMPLETED
      44 A5   50  C2  02FE          SUBL    R0,IRP$L_OBCNT(R5)            ; BYTE COUNT REMAINING
              25  13  0302          BEQL    40$                           ; BRANCH IF NOTHING LEFT TO DO
         40 A5    D4  0304          CLRL    IRP$L_ABCNT(R5)               ; ZERO ACCUMULATED BYTE COUNT
                      0307
         30 A5    B4  0307          CLRW    IRP$W_BOFF(R5)                ; ZERO BOFF AND
         44 A5    D0  030A          MOVL    IRP$L_OBCNT(R5),-
         32 A5        030D                  IRP$L_BCNT(R5)                ; SET NEW BYTE COUNT
         48 A5    D6  030F          INCL    IRP$L_SEGVBN(R5)              ; SEGMENT VBN WAS POINTING AT ERROR VBN
      53 4C A5    D0  0312          MOVL    IRP$L_DIAGBUF(R5),R3          ; STARTING SVAPTE OF ENTIRE TRANSFER
4C A5    6341     DE  0316          MOVAL   (R3)[R1],IRP$L_DIAGBUF(R5) ; STARTING PTE ADDRESS OF THIS SEGMENT
                      031B
                      031B          .IF     DF,CA$_MEASURE_IOT
              53  DD  031B          PUSHL   R3                            ;REMEMBER SVAPTE
          53  55  D0  031D          MOVL    R5,R3                         ;SET ADR OF IRP
        FCDD'     30  0320          BSBW    PMS$START_RQ                  ; INSERT START OF I/O REQUEST MESSAGE
          53 8ED0     0323          POPL    R3                            ;RESTORE SVAPTE
                      0326          .ENDC
                      0326
         00EB     31  0326          BRW     QNXTSEG                       ; QUEUE THIS SEGMENT AND RETURN TO IOPOST
      50  55      D0  0329  40$:    MOVL    R5,R0                         ; I/O PACKET ADDRESS
         FF45     31  032C          BRW     PAGIO_ERR_DONE
                      032F  ;
                      032F  ; PAGE WRITE ERROR - CLEAN UP LOGIC
                      032F  ;
                      032F  ;       R3 = PTE ADDRESS FOR ERROR PAGE
                      032F  ;       R4 = PCB ADDRESS
                      032F  ;       R5 = PROCESS HEADER ADDRESS
                      032F  ;       R6 = I/O REQUEST PACKET ADDRESS
                      032F  ;       R7 = 0 IF ALL COMPLETION LOGIC IS DONE IN IOPOST
                      032F  ;          = NON-ZERO IF COMPLETION (AND ERROR REPORT) ARE TO BE
                      032F  ;            RETURNED TO THE PROCESS.
                      032F  ;
                      032F  PAGWRT_ERR1:
              57  DD  032F          PUSHL   R7                            ; SAVE KERNEL AST FLAG
              09  EF  0331          EXTZV   #VA$V_VPN,-
              17      0333                  #<32-VA$V_VPN>,-              ; PAGE COUNT TRANSFERRED
     50    3A A6      0334                  IRP$L_IOST1+2(R6),R0
              09  EF  0337          EXTZV   #VA$V_VPN,-
              17      0339                  #<32-VA$V_VPN>,-              ; ORIGINAL PAGE COUNT
     57    44 A6      033A                  IRP$L_OBCNT(R6),R7
          57  50  C2  033D          SUBL    R0,R7                         ; COUNT OF REMAINING PAGES
              6E  D5  0340          TSTL    (SP)                          ; IF NOT REPORTING ERROR TO PROCESS
              03  12  0342          BNEQ    20$
          57  01  D0  0344          MOVL    #1,R7                         ; ONLY CLEAN UP THE ERROR PAGE HERE
                      0347                                                ; REST OF TRANSFER WILL BE DONE BY PAGIO_ERR
              7E  D4  0347  20$:    CLRL    -(SP)                         ; INIT "ERROR PAGE" FLAG
  50  63  15  00  EF  0349          EXTZV   #PTE$V_PFN,#PTE$S_PFN,(R3),R0 ;GET PFN FROM PTE
  00000000'EF 50  D1  034E          CMPL    R0,MMG$GL_MAXPFN              ;IS THIS PAGE IN SHARED MEMORY?
              25  1A  0355          BGTRU   130$                          ;BR IF PAGE IN SH MEM, NO PFN DATABASE
              53  DD  0357  70$:    PUSHL   R3                            ; SAVE SVAPTE
         0028     30  0359          BSBW    PFN_IO_DONE                   ; COMPLETE I/O FOR THIS PAGE
00 0000'DF40  07  E2  035C          BBSS    #PFN$V_MODIFY,@W^PFN$AB_STATE[R0],80$ ; FORCE MODIFY BIT
                      0363  80$:
                      0363  ;
                      0363  ; CONDITION CODES STILL SET FROM DECREF AT END OF PFN_IO_DONE
```

```
                          0363   ;
             10     14    0363            BGTR    120$                        ; BRANCH IF NOT THE LAST REFERENCE
   08 04 AE  00     E2    0365            BBSS    #0,4(SP),100$               ; BRANCH IF NOT ERROR PAGE
                          036A   ;
                          036A   ; THIS IS THE PAGE THAT HAD THE WRITE ERROR
                          036A   ;
         52  02     D0    036A            MOVL    #PFN$C_BADPAGLST,R2         ; PUT IT ON THE BAD PAGE LIST
           FC90'    30    036D            BSBW    MMG$INSPFNT                 ; WITH ''MODIFY'' SET AND ''BAD'' CLEAR
             03     11    0370            BRB     120$
           FC8B'    30    0372   100$:    BSBW    MMG$RELPFN                  ; NO MORE REFERENCES, RELEASE THE PAGE
   53    8E  04     C1    0375   120$:    ADDL3   #4,(SP)+,R3                 ; NEXT PTE ADDRESS
          DB 57     F5    0379            SOBGTR  R7,70$
             03     BA    037C   130$:    POPR    #^M<R0,R1>                  ;CLEAN OFF BAD PAGE FLAG
                          037E                                                ; R1 = SAVED KERNEL AST INDICATOR
         57  51     D0    037E            MOVL    R1,R7                       ; PUT IT IN R7, SET CONDITION CODES
           FE54     31    0381            BRW     PA[illegible]_ERR_DONE
                          0384   ;
                          0384   ; PFN_IO_DONE
                          0384   ;
                          0384   ; INPUTS:
                          0384   ;
                          0384   ;        R3 = SVAPTE
                          0384   ;        R4 = PROCESS CONTROL BLOCK ADDRESS OF PROCESS THAT REQUESTED THE I/O
                          0384   ;        R5 = PROCESS HEADER OF THE PROCESS THAT REQUESTED THIS I/O
                          0384   ;        R6 = I/O REQUEST PACKET ADDRESS
                          0384   ;
                          0384   ; OUTPUTS:
                          0384   ;
                          0384   ;        R0 = PFN
                          0384   ;        R3 PRESERVED
                          0384   ;        IRP$W_BOFF(R6) INCREMENTED IF THIS WAS A COLLISION PAGE
                          0384   ;        CONDITION CODES SET FROM DECW @W^PFN$AW_REFCNT[R0]
                          0384   ;
                          0384   PFN_IO_DONE:
 50  63  15  00     EF    0384            EXTZV   #PTE$V_PFN,#PTE$S_PFN,(R3),R0 ; GET PAGE FRAME NUMBER
          E8 8F     8B    0389            BICB3   #^C<PFN$M_COLLISION ! PFN$M_PAGTYP>,- ; FETCH THESE
  51  0000'DF40           038C                    @W^PFN$AB_TYPE[R0],R1       ; BITS FROM PFN TYPE BYTE
       09 51 04     E5    0391            BBCC    #PFN$V_COLLISION,R1,20$ ; CLEAR COLLISION BIT, BRANCH IF WAS CLEAR
  0000'DF40  10     8A    0395            BICB    #PFN$M_COLLISION,@W^PFN$AB_TYPE[R0] ; CLEAR IT IN PFN DATA
          30 A6     B6    039B            INCW    IRP$W_BOFF(R6)              ; MUST EMPTY THE COLLISION QUEUE
         04  51     91    039E   20$:     CMPB    R1,#PFN$C_PPGTBL            ; IF PROCESS PAGE TABLE PAGE
             07     12    03A1            @NEQ    40$
      51  42 A5     3C    03A3            MOVZWL  PHD$W_PHVINDEX(R5),R1       ; MUST COUNT ONE LESS
          FC56'     30    03A7            BSBW    MMG$DECPHDREF1              ; PROCESS HEADER REFERENCE
                          03AA   40$:     DECREF                              ; ONE LESS REFERENCE FOR THE PAGE
             05           03B4            RSB                                 ; RETURN WITH CONDITION CODES SET
                          03B5                                                ; TO NEW STATE OF THE REFCNT
```

```
                       03B5        .SBTTL  VIRTUAL (OR LOGICAL) I/O COMPLETION
                       03B5  ;
                       03B5  ; VIRTUAL (OR LOGICAL) I/O COMPLETION
                       03B5  ;
                       03B5  ; CALLING SEQUENCE:
                       03B5  ;
                       03B5        BRW     VIRTUAL
                       03B5  ;
                       03B5  ; INPUTS:
                       03B5  ;
                       03B5        R1 = REQUESTED BYTE COUNT, POSSIBLY DIFFERENT FROM TRANSFERRED
                       03B5             BYTE COUNT FOR MAGTAPE
                       03B5        R2 = IRP$W_BOFF CONTENTS
                       03B5        R3 = SVAPTE OF START OF TRANSFER
                       03B5        R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
                       03B5        R5 = IRP ADDRESS
                       03B5  ;
                       03B5  ; OUTPUTS:
                       03B5  ;
                       03B5        BRANCHES TO UNLOCK, PRESERVING R1,R2,R3
                       03B5        OR BRANCHES TO IOPOST
                       03B5  ;
                       03B5
                       03B5        .ENABL  LSB
                       03B5
                       03B5  VIRTUAL_LOGIO:                               ; VIRTUAL (OR LOGICAL) I/O FUNCTION
          46 A5   B5   03B5        TSTW    IRP$L_OBCNT+2(R5)            ; SEE IF BYTE COUNT > 64K
             0F   13   03B8        BEQL    1$                           ; EQL IMPLIES NO, BRANCH TO OLD CODE
                       03BA
      50   3A A5  D0   03BA        MOVL    IRP$L_IOST1+2(R5), R0        ; Else pickup new, longer count.
      40 A5   50  C0   03BE        ADDL    R0, IRP$L_ABCNT(R5)          ; Accumulate total bytes transfered.
   3A A5   40 A5  D0   03C2        MOVL    IRP$L_ABCNT(R5), -           ; Set accumulated bytes transfered.
                       03C7                IRP$L_IOST1+2(R5)
             0D   11   03C7        BRB     3$                           ; Rejoin common code.
                       03C9
      50   3A A5  3C   03C9  1$:   MOVZWL  IRP$L_IOST1+2(R5), R0        ; Get old bytes transfered count.
      40 A5   50  C0   03CD        ADDL    R0, IRP$L_ABCNT(R5)          ; Accumulate total bytes transfered.
   3A A5   40 A5  B0   03D1        MOVW    IRP$L_ABCNT(R5), -           ; Set accumulated bytes transfered.
                       03D6                IRP$L_IOST1+2(R5)            ; (Note movw due to code path that
                       03D6                                             ; insures < 64K byte transfer.)
                       03D6
             50   DD   03D6  3$:   PUSHL   R0                           ; Save # bytes transferred.
          51   50 D1   03D8        CMPL    R0, R1                       ; Do bytes xfered and requested match?
             13   13   03DB        BEQL    9$                           ; Branch if they match.
      50   1C A5  D0   03DD        MOVL    IRP$L_UCB(R5),R0             ; RO => UCB.
             05   E0   03E1        BBS     S^#DEV$V_SQD,-
        0A 38 A0       03E3                UCB$L_DEVCHAR(R0),9$         ; If SET, sequential device
        06 38 A5  E9   03E6        BLBC    IRP$L_IOST1(R5), 9$          ; If xfer count wrong, guarantee
   38 A5   2234 8F B0  03EA        MOVW    #SS$_INCSEGTRA, -            ; that final status is an error
                       03F0                IRP$L_IOST1(R5)              ; (either the driver's or ours).
   50   8E   F7 8F 78  03F0  9$:   ASHL    #-VA$S_BYTE,(SP)+, R0        ; Calculate number of blocks transfered.
      48 A5   50  C0   03F5        ADDL    R0, IRP$L_SEGVBN(R5)         ; Calculate next disk segment address.
        4C 38 A5  E9   03F9        BLBC    IRP$L_IOST1(R5),20$          ; IF LBC I/O ERROR
      50   1C A5  D0   03FD        MOVL    IRP$L_UCB(R5),R0             ; GET ADDRESS OF DEVICE UCB
   2E 38 A0   05  E0   0401        BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R0),10$ ; IF SET, SEQUENTIAL DEVICE
          40 A5   C3   0406        SUBL3   IRP$L_ABCNT(R5),-
          44 A5        0409                IRP$L_OBCNT(R5),-            ; CALCULATE BYTES REMAINING
```

```
          32 A5       040B                  IRP$L_BCNT(R5)             ;
             25   13  040D          BEQL    10$                        ; IF EQL NONE
 51   51  F7 8F   78  040F          ASHL    #-VA$S_BYTE,R1,R1          ; CALCULATE NUMBER OF PAGES REQUESTED
                      0414  QNXTSEG:
                      0414  ;
                      0414  ; ADVANCE THE SVAPTE TO POINT TO THE PORTION OF THE PAGE TABLES THAT MAP THE
                      0414  ; BUFFER FOR THIS SEGMENT.  IF THIS IS AN ERASE I/O, DO NOT ADVANCE THE
                      0414  ; SVAPTE, AS THE ENTIRE TRANSFER IS MAPPED BY A SINGLE PAGE TABLE PAGE.
                      0414  ;
         0000'CF  D6  0414          INCL    W^PMS$GL_SPLIT             ; COUNT A SPLIT TRANSFER
              0A  E1  0418          BBC     #IO$V_ERASE,-              ; BRANCH IF NOT ERASE - UPDATE SVAPTE
         06 20 A5     041A                  IRP$W_FUNC(R5),13$         ;
            02F9  30  041D          BSBW    CHECK_ERASE                ; IS THIS AN ERASE I/O REQUEST
           05 50  E8  0420          BLBS    R0,69$                     ; BRANCH IF YES - DO NOT ADVANCE SVAPTE
   2C A5    6341  DE  0423  13$:    MOVAL   (R3)[R1],IRP$L_SVAPTE(R5)  ; SET ADDRESS OF NEXT PTE ENTRY
      53      55  D0  0428  69$:    MOVL    R5,R3                      ; COPY I/O REQUEST PACKET ADDRESS
      55   1C A3  D0  042B          MOVL    IRP$L_UCB(R3),R5           ; COPY UCB ADDRESS
              47  10  042F          BSBB    IOC$QNXTSEG                ; QUEUE THE NEXT VIRTUAL SEGMNET
            FBD9  31  0431  5$:     BRW     IOPOST
                      0434  ;
                      0434  ; ALL SEGMENTS OF THIS TRANSFER ARE COMPLETE
                      0434  ;
                      0434  10$:
      51   44 A5  D0  0434          MOVL    IRP$L_OBCNT(R5),R1         ; GET ORIGINAL BYTE COUNT
      53   4C A5  D0  0438          MOVL    IRP$L_DIAGBUF(R5),R3       ; GET ORIGINAL PAGE TABLE ADDRESS
              04  12  043C          BNEQ    15$                        ; NEQ implies IRP$L_DIAGBUF was valid.
      53   2C A5  D0  043E          MOVL    IRP$L_SVAPTE(R5),R3        ; If not valid, then IRP$L_SVAPTE is.
   2C A5      53  D0  0442  15$:    MOVL    R3,IRP$L_SVAPTE(R5)        ; SVAPTE MUST BE CORRECT
            FC4D  31  0446          BRW     UNLOCK                     ;
                      0449
                      0449  ;
                      0449  ; I/O OPERATION ENDED WITH AN UNSUCCESSFUL STATUS
                      0449  ;
                      0449  ;               IF THE REQUEST IS  LOGICAL I/O, BRANCH BACK TO UNLOCK. (10$)
                      0449  ;
                      0449  ;               IF THE DEVICE IS A SEQUENTIAL DEVICE, THEN THE I/O PACKET IS
                      0449  ;               MERELY SENT TO THE ACP FOR NOTIFICATION OF THE ERROR.
                      0449  ;
                      0449  ;               IF THE DEVICE IS A RANDOM DEVICE, THEN THE VIRTUAL BLOCK NUMBER
                      0449  ;               STORED IN IRP$L_SEGVBN IS THE BLOCK THAT HAS AN ERROR.
                      0449  ;
                      0449
              04  E1  0449  20$:    BBC     #IRP$V_VIRTUAL,-
         E6 2A A5     044B                  IRP$W_STS(R5),10$          ; Branch If Logical I/O
           46 A5  B5  044E          TSTW    IRP$L_OBCNT+2(R5)          ; SEE IF BYTE COUNT > 64K
              05  13  0451          BEQL    30$                        ; EQL implies < 64K.
           3A A5  D4  0453          CLRL    IRP$L_IOST1+2(R5)          ; Zero byte count before recycleing IRP
              03  11  0456          BRB     40$                        ; Branch around
                      0458  30$:
           3A A5  B4  0458          CLRW    IRP$L_IOST1+2(R5)          ; Zero byte count before recycleing IRP
                      045B  40$:
      53      55  D0  045B          MOVL    R5,R3                      ; COPY IRP ADDRESS
           3E A4  B7  045E          DECW    PCB$W_DIOCNT(R4)           ; ADJUST DIRECT I/O COUNT
   2A A3      10  AA  0461          BICW    #IRP$M_VIRTUAL,IRP$W_STS(R3)  ; CLEAR VIRTUAL I/O FLAG
 2C A3     4C A3  D0  0465          MOVL    IRP$L_DIAGBUF(R3),IRP$L_SVAPTE(R3)  ; RESET PAGE TABLE ADDRESS
      52   44 A3  D0  046A          MOVL    IRP$L_OBCNT(R3),R2         ; GET ORIGINAL BYTE COUNT
            009F  30  046E          BSBW    IOC$QTOACP                 ; QUEUE PACKET TO ACP
```

```
BE   11 0471   899        BRB     5$
        0473   900
        0473   901        .DSABL  LSB
```

```
                              0473          .SBTTL  QUEUE NEXT SEGMENT
                              0473    ;
                              0473    ; FUNCTIONAL DESCRIPTION:
                              0473    ;
                              0473    ;       IOC$QNXTSEG PERFORMS THE FUNCTION OF QUEUEING THE NEXT
                              0473    ;       SEGMENT OF A VIRTUAL I/O REQUEST THAT DID NOT MAP TO A
                              0473    ;       SINGLE CONTIGUOUS I/O REQUEST.
                              0473    ;
                              0473    ; CALLING SEQUENCE:
                              0473    ;
                              0473    ;       BSBW    IOC$QNXTSEG
                              0473    ;
                              0473    ; INPUTS:
                              0473    ;
                              0473    ;       R3 = I/O REQUEST PACKET ADDRESS
                              0473    ;       R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
                              0473    ;       R5 = UCB ADDRESS OF THE ASSOCIATED DEVICE
                              0473    ;
                              0473    ; OUTPUTS:
                              0473    ;
                              0473    ;       R4 NOT PRESERVED
                              0473    ;
                              0473
                              0473          .ENABLE LSB
                              0473                                          ; Out of line code for Logical I/O.
                              0473                                          ; This code mimics results of
                              0473                                          ;  IOC$MAPVBLK for Logical I/O.
            51    50    D0    0473    5$:   MOVL    R0,R1                   ; Namely R1 = LBN.
                  24    11    0476          BRB     10$                     ; Branch back to common code.
                              0478
                              0478    IOC$QNXTSEG::
         52    18 A3    D0    0478          MOVL    IRP$L_WIND(R3),R2       ; GET ADDRESS OF MAPPING WINDOW
         51    32 A3    D0    047C          MOVL    IRP$L_BCNT(R3),R1       ; GET SIZE OF NEXT SEGMENT
         50    48 A3    D0    0480          MOVL    IRP$L_SEGVBN(R3),R0     ; GET STARTING VIRTUAL BLOCK NUMBER
                              0484    ;
                              0484    ; ALTERNATE ENTRY TO IOC$QNXTSEG:
                              0484    ;
                              0484    ;       BSBW    IOC$QNXTSEG1
                              0484    ;
                              0484    ; ADDITIONAL INPUTS:
                              0484    ;
                              0484    ;       R0 = VIRTUAL BLOCK NUMBER OF START OF NEXT SEGMENT
                              0484    ;       R1 = DESIRED BYTE COUNT OF NEXT SEGMENT
                              0484    ;       R2 = WINDOW ADDRESS
                              0484    ;
                              0484    IOC$QNXTSEG1::
            3E A4       B7    0484          DECW    PCB$W_DIOCNT(R4)        ; ADJUST THE DIRECT I/O COUNT
               04       E1    0487          BBC     #IRP$V_VIRTUAL,-        ; Branch to out of line code if this
         E7 2A A3             0489                  IRP$W_STS(R3),5$        ;  is Logical I/O.
       00000000'GF      16    048C          JSB     G^IOC$MAPVBLK           ; MAP VIRTUAL TO LOGICAL BLOCK
            1C A3    55 D0    0492          MOVL    R5,IRP$L_UCB(R3)        ; STORE POSSIBLY MODIFIED UCB ADDRESS
            32 A3    52 C2    0496          SUBL    R2,IRP$L_BCNT(R3)       ; CALCULATE SIZE OF NEXT SEGMENT
                     74 13    049A          BEQL    30$                     ; IF EQL TOTAL MAP FAILURE
      52    00B4 C5     D0    049C    10$:  MOVL    UCB$L_MAXBCNT(R5),R2    ; R2 = 0 or Max. permissible BCNT.
                     05 12    04A1          BNEQ    15$                     ; NEQ implies Max. permissible BCNT in R0.
      52    FE00 8F     3C    04A3          MOVZWL  #512*127,R2             ; If 0, use default Max. permissible.
                     0A E1    04A8    15$:  BBC     #IO$V_ERASE,-           ; BRANCH IF DEFINITELY NOT AN ERASE
```

```
         20 20 A3      04AA               IRP$W_FUNC(R3),17$              ;
               55  DD  04AD        PUSHL  R5                              ; SAVE UCB ADDRESS
         55    53  D0  04AF        MOVL   R3,R5                           ; COPY IRP ADDRESS
             0264  30  04B2        BSBW   CHECK_ERASE                     ; IS THIS AN ERASE FUNCTION?
             55 8ED0   04B5        POPL   R5                              ; RESTORE UCB ADDRESS
            12 50  E9  04B8        BLBC   R0,17$                          ; BRANCH IF IT IS NOT AN ERASE
            2C A3  D5  04BB        TSTL   IRP$L_SVAPTE(R3)                ; ARE WE USING A DUMMY PAGE TABLE?
               0D  13  04BE        BEQL   17$                             ; BRANCH IF NOT
     50   FE00 8F  3C  04C0        MOVZWL #512*127,R0                     ; GET MAX BYTE COUNT FOR PPT
         50    52  D1  04C5        CMPL   R2,R0                           ; CHECK LIMIT AGAINST MAX
               03  1B  04C8        BLEQU  17$                             ; BRANCH IF OK
         52    50  D0  04CA        MOVL   R0,R2                           ; LIMIT TRANSFER TO PPT SIZE
                       04CD
      32 A3    52  D1  04CD 17$:   CMPL   R2,IRP$L_BCNT(R3)               ; See if BCNT too large.
               04  1E  04D1        BGEQU  20$                             ; GEQU implies we are OK.
      32 A3    52  D0  04D3        MOVL   R2,IRP$L_BCNT(R3)               ; Else scale down to maximum allowed.
                       04D7 20$:
      52    32 A3  D0  04D7        MOVL   IRP$L_BCNT(R3),R2               ; GET TRANSFER BYTE COUNT
               52  D7  04DB        DECL   R2                              ; ROUND DOWN AND...
   52   52  F7 8F  78  04DD        ASHL   #-VA$S_BYTE,R2,R2               ; SHIFT DOWN FOR BLOCK COUNT - 1
         52    51  C0  04E2        ADDL   R1,R2                           ; COMPUTE ENDING BLOCK NUMBER
               13  1F  04E5        BCS    25$                             ; BRANCH ON OVERFLOW
     00B0 C5   52  D1  04E7        CMPL   R2,UCB$L_MAXBLOCK(R5)           ; AND CHECK AGAINST DEVICE SIZE
               0C  1E  04EC        BGEQU  25$                             ; BRANCH IF NOT LEGAL
         50    51  D0  04EE        MOVL   R1,R0                           ; COPY STARTING LOGICAL BLOCK NUMBER
     00000000'GF   16  04F1        JSB    G^IOC$CVTLOGPHY                 ; CONVERT LOGICAL TO PHYSICAL BLOCK
            FB06'  31  04F7        BRW    EXE$INSIOQ                      ; INSERT I/O PACKET IN DEVICE QUEUE
                       04FA                                               ; AND RETURN
                       04FA ;
                       04FA ; TO HERE IF THE VIRTUAL BLOCKS MAP OFF THE END OF THE VOLUME. COMPLETE THE
                       04FA ; I/O WITH AN ERROR. WE QUEUE THE PACKET FOR PROCESSING, RATHER THAN WANDERING
                       04FA ; OFF INTO THE COMPLETION CODE BECAUSE THIS IS A GENERALLY CALLABLE ROUTINE.
                       04FA ;
  38 A3   00DC 8F  3C  04FA 25$:   MOVZWL #SS$_ILLBLKNUM,IRP$L_IOST1(R3) ; SET ILLEGAL BLOCK NUMBER STATUS
            3C A3  D4  0500        CLRL   IRP$L_IOST2(R3)                 ; ZERO 2ND I/O STATUS LONGWORD
 00000000'FF   63  0E  0503        INSQUE (R3),@IOC$GL_PSBL               ; INSERT AT TAIL OF I/O POST QUEUE
               03  12  050A        BNEQ   26$                             ; BRANCH IF NOT EMPTY
                       050C        SOFTINT #IPL$_IOPOST                   ; WAKE UP I/O COMPLETION
                   05  050F 26$:   RSB
                       0510
                       0510 30$:
                       0510               .DISABLE LSB
                       0510 ;
                       0510 ; ALTERNATE ENTRY TO IOC$WAKACP:
                       0510 ;
                       0510               BSBW    IOC$QTOACP
                       0510 ;
                       0510 ; INPUTS:
                       0510 ;
                       0510               R2 = DESIRED BYTE COUNT
                       0510               R3 = IRP ADDRESS
                       0510               PCB$W_DIOCNT(R4) ALREADY DECREMENTED
                       0510 ;
                       0510 IOC$QTOACP:
      32 A3    52  D0  0510        MOVL   R2,IRP$L_BCNT(R3)               ; SET REMAINING BYTES TO TRANSFER
      52    18 A3  D0  0514        MOVL   IRP$L_WIND(R3),R2               ; GET WINDOW ADDRESS
      0B A2    02  E0  0518        BBS    #WCB$V_NOTFCP,WCB$B_ACCESS(R2),- ; IF SET THEN
```

```
            4E        051C              NOTFCPWCB                       ; NOT FCP WINDOW
52    1C A3    D0     051D        MOVL    IRP$L_UCB(R3),R2              ; GET ADDRESS OF DEVICE UCB
                      0521   ;
                      0521   ; FUNCTIONAL DESCRIPTION:
                      0521   ;
                      0521   ;       SUBROUTINE TO QUEUE AN I/O PACKET FOR AN ACP PROCESS AND WAKE
                      0521   ;       THE PROCESS IF ITS QUEUE WAS PREVIOUSLY EMPTY.
                      0521   ;
                      0521   ; CALLING SEQUENCE:
                      0521   ;
                      0521   ;       BSBW    IOC$WAKACP
                      0521   ;
                      0521   ; INPUTS:
                      0521   ;
                      0521   ;       R2 = DEVICE UCB ADDRESS
                      0521   ;       R3 = I/O REQUEST PACKET ADDRESS
                      0521   ;
                      0521   ; OUTPUTS:
                      0521   ;
                      0521   ;       R4 ALTERED
                      0521   ;
                      0521        .ENABL  LSB
                      0521   IOC$WAKACP::                                ; QUEUE I/O PACKET AND WAKE ACP PROCESS
                      0521        DSBINT  #IPL$_SYNCH                    ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
52    34 A2    D0     0527        MOVL    UCB$L_VCB(R2),R2               ; GET ASSOCIATED VCB ADDRESS
52    10 A2    D0     052B        MOVL    VCB$L_AQB(R2),R2               ; GET ACP QUEUE BLOCK ADDRESS
      0C A2    D5     052F        TSTL    AQB$L_ACPPID(R2)               ; PROCEDURE BASED? NO PID IF SO
         17    13     0532        BEQL    XQP                            ; EQL THEN IS NOT AN ACP
      FAC9'    30     0534        BSBW    EXE$INSERTIRP                  ; INSERT I/O PACKET IN ACP QUEUE
         0E    12     0537        BNEQ    10$                            ; IF NEQ NOT FIRST IN QUEUE
51    0C A2    D0     0539        MOVL    AQB$L_ACPPID(R2),R1            ; GET ACP PROCESS ID
      FAC0'    30     053D        BSBW    SCH$WAKE                       ; WAKE ACP PROCESS
      04 50    E8     0540        BLBS    R0,10$                         ; IF LBS ACP STILL PRESENT
                      0543        BUG_CHECK NONEXSTACP                   ; NONEXISTENT ACP PROCESS
                      0547   10$: ENBINT                                 ; RESTORE SAVED IPL
               05     054A        RSB                                    ;
                      054B   ;
                      054B   ; THIS VOLUME IS BEING HANDLED BY AN XQP INSTEAD OF AN ACP.  CALL THE
                      054B   ; XQP QUEUEING ROUTINE AS A SPECIAL KERNEL AST TO GET IN THE CONTEXT
                      054B   ; OF THE PROCESS THAT INITIATED THIS REQUEST TO HANDLE IT.
                      054B   ;
                      054B
                      054B   XQP:
         55    DD     054B        PUSHL   R5                             ; PRESERVE R5.
55    60 A3    9E     054D        MOVAB   IRP$L_FQFL(R3), R5             ; GET TEMP ACB ADDR INTO R5.
0B A5 80 8F    90     0551        MOVB    #ACB$M_KAST, ACB$B_RMOD(R5)    ; NOTE AS SPECIAL KERNEL AST
0C A5 0C A3    D0     0556        MOVL    IRP$L_PID(R3), ACB$L_PID(R5)   ; COPY PID OF PROCESS.
18 A5 0000'CF  9E     055B        MOVAB   W^EXE$QXQPPKT, ACB$L_KAST(R5)  ; ADDR OF QUEUEING ROUTINE.
         52    D4     0561        CLRL    R2                             ; NO PRIORITY INCREMENT.
      FA9A'    30     0563        BSBW    SCH$QAST                       ; QUEUE THE AST.
         55  8ED0     0566        POPL    R5                             ; RESTORE R5.
         DC    11     0569        BRB     10$                            ; BRANCH TO EXIT.
                      056B
                      056B        .DSABL  LSB
                      056B   ;
                      056B   ; WINDOW IS NOT AN FCP WINDOW, ONLY USED FOR BOOT TIME INTIALIZED WINDOWS
                      056B   ; FOR CONTIGUOUS FILES.  IT IS NOT POSSIBLE TO NEED TO TURN SUCH A WINDOW.
```

```
056B  1074 ;
056B  1075 NOTFCPWCB:
056B  1076          BUG_CHECK NOTFCPWCB,FATAL
```

```
                                  056F        .SBTTL  BUFFERED READ COMPLETION AST ROUTINE
                                  056F  ;++
                                  056F  ; FUNCTIONAL DESCRIPTION:
                                  056F  ;
                                  056F  ;       BUFPOST PERFORMS ALL NECESSARY COMPLETION OPERATIONS REQUIRED
                                  056F  ;       FOR A BUFFERED READ OPERATION IN THE CONTEXT OF THE PROCESS
                                  056F  ;       ISSUING THE I/O REQUEST.
                                  056F  ;
                                  056F  ; CALLING SEQUENCE:
                                  056F  ;
                                  056F  ;       JSB     BUFPOST
                                  056F  ;
                                  056F  ; INPUT PARAMETERS:
                                  056F  ;
                                  056F  ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                  056F  ;       R5 = IRP/AST CONTROL BLOCK.
                                  056F  ;
                                  056F  ; IMPLICIT INPUTS:
                                  056F  ;
                                  056F  ;       SCH$GL_CURPCB - POINTER TO PCB OF CURRENT PROCESS
                                  056F  ;--
                                  056F
                                  056F  BUFPOST:                                        ; BUFFERED READ COMPLETION
                  00E0 8F    BB   056F          PUSHR   #^M<R5,R6,R7>                   ; SAVE REGISTERS
              56    2C A5    D0   0573          MOVL    IRP$L_SVAPTE(R5),R6             ; GET ADDRESS OF I/O BUFFER
              57    32 A5    D0   0577          MOVL    IRP$L_BCNT(R5),R7               ; GET COUNT OF BYTES OR DESCRIPTORS
           4B 2A A5    03    E1   057B          BBC     #IRP$V_COMPLX,IRP$W_STS(R5),40$ ;IF CLR, NOT COMPLEX BUFFER FORMAT
           59 2A A5    05    E0   0580          BBS     #IRP$V_CHAINED,IRP$W_STS(R5),50$ ;IF SET ,CHAINED BUFFERS
              56       66    D0   0585          MOVL    (R6),R6                         ; GET ADDRESS OF FIRST BUFFER DESCRIPTOR
              50    02 A6    3C   0588  10$:    MOVZWL  2(R6),R0                        ; GET COUNT OF BYTES TO TRANSFER
                       32    13   058C          BEQL    30$                             ; IF EQL NONE THIS DESCRIPTOR
              51    04 A6    D0   058E          MOVL    4(R6),R1                        ; GET ADDRESS OF USER BUFFER
              50       51    C0   0592          ADDL    R1,R0                           ; CALCULATE ENDING ADDRESS OF BUFFER
              51  01FF 8F    AA   0595          BICW    #VA$M_BYTE,R1                   ; TRUNCATE ADDRESS TO PAGE BOUNDARY
              50       51    C2   059A          SUBL    R1,R0                           ; COMPUTE NUMBER OF BYTES TO PROBE
              54       66    3C   059D          MOVZWL  (R6),R4                         ; GET OFFSET TO DATA AREA
              53  FE00 8F    32   05A0          CVTWL   #-^X200,R3                      ; SET ADDITION CONSTANT
                                  05A5  20$:    IFNOWRT R0,(R1),35$,(R6)[R4]            ; CAN BUFFER BE WRITTEN?
              51       53    C2   05AC          SUBL    R3,R1                           ; UPDATE ADDRESS OF BUFFER
              50     6043    3E   05AF          MOVAW   (R0)[R3],R0                     ; UPDATE REMAINING LENGTH
                       F0    14   05B3          BGTR    20$                             ; IF GEQ MORE TO CHECK
  04 B6    01 A644  02 A6    28   05B5          MOVC    2(R6),1(R6)[R4],@4(R6)          ; MOVE DATA TO USER BUFFER
              55       6E    D0   05BD          MOVL    (SP),R5                         ; RESTORE ADDRESS OF I/O PACKET
              56       08    C0   05C0  30$:    ADDL    #8,R6                           ; ADVANCE TO NEXT BUFFER DESCRIPTOR
                    C2 57    F5   05C3          SOBGTR  R7,10$                          ; ANY MORE DESCRIPTORS TO PROCESS?
                     007D    31   05C6          BRW     130$                            ;
                       78    11   05C9  35$:    BRB     120$                            ; CONTINUE
                     017E    30   05CB  40$:    BSBW    MOVBUF                          ; MOVE BUFFER TO USER
              55       6E    D0   05CE          MOVL    (SP),R5                         ; RETRIEVE ADDRESS OF I/O PACKET
           70 2A A5    0A    E1   05D1          BBC     #IRP$V_MBXIO,IRP$W_STS(R5),130$; BR IF NOT MAILBOX READ
              50       02    9A   05D6          MOVZBL  #RSN$_MAILBOX,R0                ; SET UP RESOURCE RELEASE
                    FA24'    30   05D9          BSBW    SCH$RAVAIL                      ; DECLARE MAILBOX RESOURCE AVAILABLE
                       68    11   05DC          BRB     130$                            ;
                                  05DE
                                  05DE  ;
                                  05DE  ; NB: THE FOLLOWING SECTION OF CODE USES A WORD-SIZE BUFFER LENGTH
                                  05DE  ;     (ALTHOUGH IRP$L_BCNT WAS EXPANDED TO BE A LONGWORD).
```

```
                        05DE      ;
           50    57  D0 05DE  50$:    MOVL    R7,R0                    ; SET LENGTH OF USER BUFFER
     51    04    A6  D0 05E1          MOVL    4(R6),R1                 ; SET ADDRESS OF USER BUFFER
           50    51  C0 05E5          ADDL    R1,R0                    ; CALCULATE END OF USER BUFFER
     51  01FF    8F  AA 05E8          BICW    #VA$M_BYTE,R1            ; TRUNCATE TO PAGE BOUNDARY
           50    51  C2 05ED          SUBL    R1,R0                    ; COMPUTE NUMBER OF BYTES TO PROBE
52    0B A5    02    00  EF 05F0          EXTZV   #0,#2,IRP$B_RMOD(R5),R2  ; GET REQUEST ACCESS MODE
         53  FE00    8F  32 05F6          CVTWL   #-^X200,R3               ; SET ADDITION CONSTANT
                        05FB  60$:    IFNOWRT R0,(R1),90$,R2       ; CAN BUFFER BE WRITTEN?
           51    53  C2 0601          SUBL    R3,R1                    ; UPDATE ADDRESS OF BUFFER
        50     6043  3E 0604          MOVAW   (R0)[R3],R0              ; CALCULATE NEW LENGTH
                 F1  14 0608          BGTR    60$                      ; IF GEQ MORE TO PROBE
     53    04    A6  D0 060A          MOVL    4(R6),R3                 ; GET STARTING ADDRESS OF USER BUFFER
     0C A6       57  B1 060E  70$:    CMPW    R7,CXB$W_LENGTH(R6)      ; REMAINING LENGTH LARGER THAN DATA AREA?
                 04  1E 0612          BGEQU   80$                      ; IF GEQU YES
     0C A6       57  B0 0614          MOVW    R7,CXB$W_LENGTH(R6)      ; TRUNCATE LENGTH OF DATA AREA
63    96    0C   A6  28 0618  80$:    MOVC    CXB$W_LENGTH(R6),@(R6)+,(R3) ; MOVE DATA TO USER BUFFER
           55    6E  D0 061D          MOVL    (SP),R5                  ; RETRIEVE ADDRESS OF I/O PACKET
     57    08    A6  A2 0620          SUBW    CXB$W_LENGTH-4(R6),R7    ; REDUCE REMAINING BYTES TO TRANSFER
                 0B  13 0624          BEQL    100$                     ; IF EQL DONE
     56    0C    A6  D0 0626          MOVL    CXB$L_LINK-4(R6),R6      ; GET ADDRESS OF NEXT BUFFER IN CHAIN
                 E2  12 062A          BNEQ    70$                      ; IF NEQ MORE TO GO
                 03  11 062C          BRB     100$                     ;
               0138  30 062E  90$:    BSBW    ACCVIO                   ; SET ACCESS VIOLATION
     50    2C    A5  D0 0631  100$:   MOVL    IRP$L_SVAPTE(R5),R0      ; GET ADDRESS OF FIRST BUFFER
     56    10    A0  D0 0635  110$:   MOVL    CXB$L_LINK(R0),R6        ; GET ADDRESS OF NEXT BUFFER
                 0F  13 0639          BEQL    140$                     ; IF EQL NONE
              F9C2'  30 063B          BSBW    EXE$DEANONPAGED          ; DEALLOCATE BUFFER
           50    56  D0 063E          MOVL    R6,R0                    ; SET ADDRESS OF NEXT BUFFER
                 F2  11 0641          BRB     110$                     ;
               0123  30 0643  120$:   BSBW    ACCVIO                   ; SET ACCESS VIOLATION STATUS
     50    2C    A5  D0 0646  130$:   MOVL    IRP$L_SVAPTE(R5),R0      ; GET ADDRESS OF BUFFER TO RELEASE
              F9B3'  30 064A  140$:   BSBW    EXE$DEANONPAGED          ; DEALLOCATE BUFFER
         00E0    8F  BA 064D          POPR    #^M<R5,R6,R7>            ; RESTORE REGISTERS
```

```
                              0651  .SBTTL  DIRECT I/O COMPLETION AST ROUTINE
                              0651 ;++
                              0651 ; FUNCTIONAL DESCRIPTION:
                              0651 ;
                              0651 ;       DIRPOST PERFORMS ALL GENERAL I/O COMPLETION ACTIVITIES WHICH
                              0651 ;       MUST BE DONE IN THE CONTEXT OF THE PROCESS. THESE INCLUDE
                              0651 ;       I/O STATUS POSTING IF AN IOSB WAS SPECIFIED, CHANNEL CONTROL
                              0651 ;       BLOCK ACTIVITY COUNT DECREMENTING, QUEUEING OF ANY REQUESTED
                              0651 ;       AST OR RELEASE OF THE I/O REQUEST PACKET.
                              0651 ;
                              0651 ; CALLING SEQUENCE:
                              0651 ;
                              0651 ;       JSB     DIRPOST
                              0651 ;
                              0651 ; INPUT PARAMETERS:
                              0651 ;
                              0651 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                              0651 ;       R5 = IRP/AST CONTROL BLOCK ADDRESS.
                              0651 ;
                              0651 ; IMPLICIT INPUTS:
                              0651 ;
                              0651 ;       SCH$GL_CURPCB - POINTER TO CURRENT PCB
                              0651 ;--
                              0651
                              0651 DIRPOST:                                    ; DIRECT I/O POSTING AST
50   2A A5   01   00    EF    0651         EXTZV   #IRP$V_BUFIO,#1,IRP$W_STS(R5),R0 ; GET INDEX TO ACCOUNTING ENTRY
   51   00000000'9F     D0    0657         MOVL    @#CTL$GL_PHD,R1            ; GET PROCESS HEADER ADDRESS
              54 A140   D6    065E         INCL    PHD$L_DIOCNT(R1)[R0]       ; ACCOUNT FOR BUFFERED OR DIRECT I/O
                              0662
            00000002          0662         .IF NE  CA$_MEASURE                ; CHECK FOR MEASUREMENT ENABLED
        00000000'EF40   D6    0662         INCL    PMS$GL_DIRIO[R0]           ; UPDATE MEASUREMENT I/O COUNTER
                              0669         .ENDC
                              0669
      1D 2A A5   07     E1    0669         BBC     #IRP$V_DIAGBUF,IRP$W_STS(R5),10$ ; IF CLR, NO DIAGNOSTIC BUFFER
             00E0 8F    BB    066E         PUSHR   #^M<R5,R6,R7>              ; SAVE REGISTERS
          56   4C A5    D0    0672         MOVL    IRP$L_DIAGBUF(R5),R6       ; GET ADDRESS OF DIAGNOSTIC BUFFER
          57   08 A6    3C    0676         MOVZWL  IRP$W_SIZE(R6),R7          ; GET SIZE OF DIAGNOSTIC BUFFER
             57   0C    C2    067A         SUBL    #12,R7                     ; REDUCE BY SIZE OF BUFFER HEADER
                00CC    30    067D         BSBW    MOVBUF                     ; MOVE DIAGNOSTIC INFORMATION TO USER
             00E0 8F    BA    0680         POPR    #^M<R5,R6,R7>              ; RESTORE REGISTERS
          50   4C A5    D0    0684         MOVL    IRP$L_DIAGBUF(R5),R0       ; RETRIEVE ADDRESS OF DIAGNOSTIC BUFFER
               F975'    30    0688         BSBW    EXE$DEANONPAGED            ; DEALLOCATE DIAGNOSTIC BUFFER
          50   28 A5    32    068B 10$:    CVTWL   IRP$W_CHAN(R5),R0          ; GET CHANNEL NUMBER (NEGATED)
   51   00000000'FF40   9E    068F         MOVAB   @CTL$GL_CCBBASE[R0],R1     ; SET CCB BASE ADDRESS
               0A A1    B7    0697         DECW    CCB$W_IOC(R1)              ; DECREMENT I/O COUNT FOR CHANNEL
                  13    12    069A         BNEQ    30$                        ; NOT IDLE YET
          53   0C A1    D0    069C         MOVL    CCB$L_DIRP(R1),R3          ; GET ADDRESS OF DEACCESS PACKET
                  0D    13    06A0         BEQL    30$                        ; IF EQL NONE
               0C A1    D4    06A2         CLRL    CCB$L_DIRP(R1)             ; CLEAR ADDRESS OF DEACCESS PACKET
               0A A1    B6    06A5         INCW    CCB$W_IOC(R1)              ; ACCOUNT FOR DEACCESS
          52   1C A3    D0    06A8         MOVL    IRP$L_UCB(R3),R2           ; GET ASSIGNED DEVICE UCB ADDRESS
                FE72    30    06AC         BSBW    IOC$WAKACP                 ; QUEUE I/O PACKET AND WAKE ACP
                              06AF                                            ; R4 ALTERERED
                              06AF 30$:
                              06AF ;
                              06AF ; R4 DOES NOT NECESSARILY HAVE CURRENT PCB ADDRESS IN IT AT THIS POINT
                              06AF ;
```

```
                              06AF   IOC$DIRPOST1::
            50    24 A5   D0  06AF          MOVL    IRP$L_IOSB(R5),R0           ; GET IOSB ADDRESS
                     10   13  06B3          BEQL    35$                         ; IF EQL NONE SPECIFIED
   51    0B A5    02      00  EF  06B5      EXTZV   #0,#2,IRP$B_RMOD(R5),R1     ; GET REQUEST ACCESS MODE
                              06BB          IFNOWRT #8,(R0),35$,R1              ; CAN I/O STATUS BE WRITTEN?
            60    38 A5   7D  06C1          MOVQ    IRP$L_IOST1(R5),(R0)        ; MOVE STATUS INTO IOSB
            51    0C A5   D0  06C5   35$:   MOVL    IRP$L_PID(R5),R1            ; PROCESS IDENTIFICATION
         17 2A A5     0B  E0  06C9          BBS     #IRP$V_EXTEND,IRP$W_STS(R5),50$ ; BRANCH TO DEALLOCATE IRPE'S
         05 0B A5     06  E1  06CE   37$:   BBC     #ACB$V_QUOTA,IRP$B_RMOD(R5),40$ ; IF CLR, NO AST SPECIFIED
                     52   D4  06D3          CLRL    R2                          ; SET NULL PRIORITY INCREMENT
                  F928'   31  06D5          BRW     SCH$QAST                    ; QUEUE AST FOR REQUESTOR
                     0A   E1  06D8   40$:   BBC     #IO$V_ERASE,-               ; BRANCH IF NOT AN ERASE REQUEST
               02 20 A5       06DA          IRP$W_FUNC(R5),42$                  ;
                     1F   10  06DD          BSBB    CLEANUP_ERASE               ; CLEAN UP AFTER AN ERASE REQUEST
            50       55   D0  06DF   42$:   MOVL    R5,R0                       ; SETUP ADDRESS FOR DEALLOCATE
                  F91B'   31  06E2          BRW     EXE$DEANONPAGED             ; AND RELEASE I/O PACKET
                              06E5
                              06E5   ;
                              06E5   ; DEALLOCATE IRPE'S
                              06E5   ;
                              06E5
            50    54 A5   D0  06E5   50$:   MOVL    IRP$L_EXTEND(R5),R0         ; GET ADDRESS OF FIRST IRPE
                              06E9
                     54   D4  06E9   60$:   CLRL    R4                          ; WILL HOLD ADDRESS OF NEXT IRPE
         04 2A A0     0B  E1  06EB          BBC     #IRPE$V_EXTEND,IRPE$W_STS(R0),70$ ; BR. IF NO MORE IRPE'S
            54    54 A0   D0  06F0          MOVL    IRPE$L_EXTEND(R0),R4        ; SAVE ADDRESS OF NEXT IRPE
                  F909'   30  06F4   70$:   BSBW    EXE$DEANONPAGED             ; DEALLOCATE IRPE POINTED TO BY R0
            50       54   D0  06F7          MOVL    R4,R0                       ; PUT ADDRESS OF NEXT IRPE IN R0
                     ED   12  06FA          BNEQ    60$                         ; BR. IF THERE IS ANOTHER IRPE
                     D0   11  06FC          BRB     37$                         ; DONE DEALLOCATING IRPE'S
```

```
                                 06FE          .SBTTL  ERASE I/O HELPER ROUTINES
                                 06FE          ;++
                                 06FE          ; CLEANUP_ERASE
                                 06FE          ;
                                 06FE          ; LOCAL SUBROUTINE TO FINISH PROCESSING AN ERASE REQUEST
                                 06FE          ; THE CLEANUP WILL VARY WITH THE TYPE OF ERASE REQUEST.
                                 06FE          ; SEE THE ROUTINE HEADER OF THE SUBROUTINE "SETUP_ERASE"
                                 06FE          ; IN SYSACPFDT FOR DETAILS.
                                 06FE          ;
                                 06FE          ; INPUT:          R5 = IRP ADDRESS
                                 06FE          ; OUTPUT:         NONE.
                                 06FE          ;--
                                 06FE
                                 06FE  CLEANUP_ERASE:                                  ; FINISH UP AFTER AN ERASE REQUEST
              19      10         06FE          BSBB    CHECK_ERASE                     ; IS THIS AN ERASE I/O?
           15 50      E9         0700          BLBC    R0,69$                          ; BRANCH IF NOT
      50   2C A5      D0         0703          MOVL    IRP$L_SVAPTE(R5),R0             ; GET ADDRESS OF PPT
              0F      13         0707          BEQL    69$                             ; BRANCH IF NONE
 50    00000000'GF    D1         0709          CMPL    G^EXE$GL_ERASEPPT,R0            ; IS THIS THE SYSTEM PPT?
              06      13         0710          BEQL    69$                             ; BRANCH IF YES
        50    0C      C2         0712          SUBL2   #12,R0                          ; CALC ADDRESS OF HEADER
            F8E8'     30         0715          BSBW    EXE$DEANONPAGED                 ; RETURN THE POOL TO THE SYSTEM
                      05         0718  69$:    RSB                                     ; RETURN
                                 0719
                                 0719
                                 0719  ;++
                                 0719  ; CHECK_ERASE
                                 0719  ;
                                 0719  ; LOCAL SUBROUTINE TO DETERMINE IF THIS IRP IS FOR AN ERASE I/O REQUEST.
                                 0719  ; THIS LEVEL OF PARANOIA IS NECESSARY TO PREVENT THE TOTAL CHAOS THAT
                                 0719  ; WOULD ARISE SHOULD AN IRP THAT 'LOOKS' LIKE AN ERASE IRP BE TREATED
                                 0719  ; INCORRECTLY.
                                 0719  ;
                                 0719  ; INPUT:          R5 = IRP ADDRESS
                                 0719  ; OUTPUT:         R0 = STATUS; LOW BIT SET IMPLIES THIS IS AN ERASE IRP
                                 0719  ;--
                                 0719
                                 0719  CHECK_ERASE:                                    ; CHECK FOR ERASE I/O REQUEST
      50   1C A5      D0         0719          MOVL    IRP$L_UCB(R5),R0                ; GET UCB ADDRESS.  NOTE: LOW BIT CLEAR
              0A      E1         071D          BBC     #IO$V_ERASE,-                   ; BRANCH IF ERASE MODIFIER NOT SET
        29 20 A5                 071F                  IRP$W_FUNC(R5),13$
              00      ED         0722          CMPZV   #IRP$V_FCODE,-                  ; CHECK FUNCTION CODE
              06                 0724                  #IRP$S_FCODE,-                  ;   ONLY DISKS AND TAPES SUPPORT DSE
      15   20 A5                 0725                  IRP$W_FUNC(R5),#IO$_DSE         ;
              1E      13         0728          BEQL    11$                             ; BRANCH IF SO
              01      91         072A          CMPB    #DC$_DISK,-                     ; IS THIS A DISK DEVICE?
           40 A0                 072C                  UCB$B_DEVCLASS(R0)
              1B      12         072E          BNEQ    13$                             ; NO - THEREFORE NOT AN ERASE
              00      ED         0730          CMPZV   #IRP$V_FCODE,-                  ; CHECK FUNCTION CODE
              06                 0732                  #IRP$S_FCODE,-                  ;   ONLY DISKS SUPPORT THE MODIFIED-
           20 A5                 0733                  IRP$W_FUNC(R5),-                ;   WRITE TYPE OF ERASE REQUEST
              0B                 0735                  #IO$_WRITEPBLK                  ;
              10      13         0736          BEQL    11$
              00      ED         0738          CMPZV   #IRP$V_FCODE,-
              06                 073A                  #IRP$S_FCODE,-
           20 A5                 073B                  IRP$W_FUNC(R5),-
              20                 073D                  #IO$_WRITELBLK
```

```
            08    13  073E  1315          BEQL    11$
            00    ED  0740  1316          CMPZV   #IRP$V_FCODE,-
            06        0742  1317                  #IRP$S_FCODE,-
         20 A5        0743  1318                  IRP$W_FUNC(R5),-
            30        0745  1319                  #IO$_WRITEVBLK
            03    12  0746  1320          BNEQ    13$              ; NOT A WRITE - THEREFORE NOT AN ERASE
   50       01    88  0748  1321  11$:    BISB2   #1,R0            ; SET LOW BIT IN R0 TO INDICATE ERASE
                  05  074B  1322  13$:    RSB                      ; RETURN STATUS VALUE
```

```
                               074C          .SBTTL  MOVE DATA TO USER BUFFER
                               074C     ;
                               074C     ; SUBROUTINE TO MOVE DATA FROM A SIMPLE BUFFERED I/O BUFFER TO A USER BUFFER
                               074C     ;
                               074C
                               074C     MOVBUF:                                        ; MOVE BUFFER
              51    57   D0    074C             MOVL    R7,R1                          ; SET LENGTH OF USER BUFFER
                    17   13    074F             BEQL    5$                             ; BR IF NULL STRING
          50    04  A6   D0    0751             MOVL    4(R6),R0                       ; GET ADDRESS OF USER BUFFER
   53   0B A5   02  00   EF    0755             EXTZV   #0,#2,IRP$B_RMOD(R5),R3        ; GET REQUEST ACCESS MODE
          00000000'      EF 16 075B             JSB     EXE$PROBEW                     ; CHECK ACCESS
                05  50   E9    0761             BLBC    R0,ACCVIO                      ; IF LBC, NO ACCESS
        96    96    57   28    0764             MOVC    R7,@(R6)+,@(R6)+               ; MOVE DATA TO USER BUFFER
                         05    0768     5$:     RSB                                    ; RETURN
           38 A5    0C   B0    0769     ACCVIO: MOVW    #SS$_ACCVIO,IRP$L_IOST1(R5) ; SET FINAL TRANSFER STATUS
                         05    076D             RSB
```

```
                          076E          .SBTTL  UNLOCK AREAS IN IRPE'S
                          076E  ;++
                          076E  ; FUNCTIONAL DESCRIPTION:
                          076E  ;
                          076E  ;       THIS ROUTINE UNLOCKS THE AREAS DESCRIBED BY FIELDS IN THE IRPE'S.  EACH
                          076E  ;       IRPE HAS SPACE TO HOLD TWO AREA DESCRIPTIONS.
                          076E  ;
                          076E  ; CALLING SEQUENCE:
                          076E  ;
                          076E  ;       BSBW    UNLOCK_MORE
                          076E  ;
                          076E  ; INPUT PARAMETERS:
                          076E  ;
                          076E  ;       R5 = I/O REQUEST PACKET ADDRESS
                          076E  ;
                          076E  ; SIDE EFFECTS:
                          076E  ;
                          076E  ;       R0 - R3 ARE NOT PRESERVED
                          076E  ;--
                          076E
                          076E          ASSUME  IRP$L_EXTEND EQ IRPE$L_EXTEND
                          076E
                          076E  UNLOCK_MORE:
             55      DD   076E          PUSHL   R5                      ; SAVE IRP ADDRESS
                          0770
                          0770  10$:    ; UNLOCK AREAS SPECIFIED IN NEXT IRPE
                          0770
      55    54 A5    D0   0770          MOVL    IRPE$L_EXTEND(R5),R5    ; GET ADDRESS OF NEXT IRPE
      53    2C A5    D0   0774          MOVL    IRPE$L_SVAPTE1(R5),R3   ; GET SVAPTE OF FIRST AREA
               0A    13   0778          BEQL    20$                     ; BR. IF NOTHING TO UNLOCK
      52    30 A5    3C   077A          MOVZWL  IRPE$W_BOFF1(R5),R2     ; GET BYTE OFFSET IN PAGE
      51    34 A5    D0   077E          MOVL    IRPE$L_BCNT1(R5),R1     ; GET SIZE OF AREA
               19    10   0782          BSBB    UNLK                    ; UNLOCK FIRST AREA
                          0784
      53    38 A5    D0   0784  20$:    MOVL    IRPE$L_SVAPTE2(R5),R3   ; GET SVAPTE OF SECOND AREA
               0A    13   0788          BEQL    30$                     ; BR. IF NOTHING TO UNLOCK
      52    3C A5    3C   078A          MOVZWL  IRPE$W_BOFF2(R5),R2     ; GET BYTE OFFSET IN PAGE
      51    40 A5    D0   078E          MOVL    IRPE$L_BCNT2(R5),R1     ; GET SIZE OF AREA
               09    10   0792          BSBB    UNLK                    ; UNLOCK SECOND AREA
                          0794
 D7 2A A5      0B    E0   0794  30$:    BBS     #IRPE$V_EXTEND,IRPE$W_STS(R5),10$ ; BR. IF THERE'S ANOTHER IRPE
             55    8ED0   0799          POPL    R5                      ; RESTORE R5
                     05   079C          RSB
                          079D
                          079D          ;
                          079D          ; LOCAL SUBROUTINE TO UNLOCK PAGES
                          079D          ;
                          079D          ;       R1 = BYTE COUNT (OR SIZE OF AREA)
                          079D          ;       R2 = BYTE OFFSET IN PAGE
                          079D          ;       R3 = SVAPTE OF START OF AREA
                          079D          ;
                          079D
   51    01FF C142   9E   079D  UNLK:   MOVAB   511(R1)[R2],R1          ; COMBINE OFFSET AND SIZE AND ROUND
 51    51    F7 8F   78   07A3          ASHL    #-VA$S_BYTE,R1,R1       ; CONVERT TO NUMBER OF PAGES TO UNLOCK
             F855'   30   07A8          BSBW    MMG$UNLOCK              ; UNLOCK PAGES
                     05   07AB          RSB
                          07AC
```

```
07AC
07AC            .END
```

```
ACB$B_RMOD          = 0000000B             IRP$L_AST           = 00000010
ACB$L_KAST          = 00000018             IRP$L_ASTPRM        = 00000014
ACB$L_PID           = 0000000C             IRP$L_BCNT          = 00000032
ACB$M_KAST          = 00000080             IRP$L_DIAGBUF       = 0000004C
ACB$V_QUOTA         = 00000006             IRP$L_EXTEND        = 00000054
ACCVIO                00000769 R     02    IRP$L_FQFL          = 00000060
AQB$L_ACPPID        = 0000000C             IRP$L_IOSB          = 00000024
BRW_QNXTSEG           000000C7 R     02    IRP$L_IOST1         = 00000038
BUFIO                 00000125 R     02    IRP$L_IOST2         = 0000003C
BUFPOST               0000056F R     02    IRP$L_KEYDESC       = 0000005C
BUG$_NONEXSTACP       ********    X  02    IRP$L_OBCNT         = 00000044
BUG$_NOTFCPWCB        ********    X  02    IRP$L_PID           = 0000000C
CA$_MEASURE         = 00000002             IRP$L_SEGVBN        = 00000048
CCB$L_DIRP          = 0000000C             IRP$L_SVAPTE        = 0000002C
CCB$W_IOC           = 0000000A             IRP$L_UCB           = 0000001C
CHECK_ERASE           00000719 R     02    IRP$L_WIND          = 00000018
CLEANUP_ERASE         000006FE R     02    IRP$M_PAGIO         = 00000004
CTL$GL_CCBBASE        ********    X  02    IRP$M_SWAPIO        = 00000040
CTL$GL_PHD            ********    X  02    IRP$M_VIRTUAL       = 00000010
CXB$L_LINK          = 00000010             IRP$S_FCODE         = 00000006
CXB$W_LENGTH        = 0000000C             IRP$V_BUFIO         = 00000000
DC$_DISK            = 00000001             IRP$V_CHAINED       = 00000005
DEV$V_FOD           = 0000000E             IRP$V_COMPLX        = 00000003
DEV$V_SQD           = 00000005             IRP$V_DIAGBUF       = 00000007
DIRIO                 00000069 R     02    IRP$V_EXTEND        = 0000000B
DIRPOST               00000651 R     02    IRP$V_FCODE         = 00000000
EVT$_COLPGA           ********    X  02    IRP$V_FILACP        = 0000000C
EVT$_PFCOM            ********    X  02    IRP$V_FUNC          = 00000001
EXE$DEANONPAGED       ********    X  02    IRP$V_KEY           = 0000000F
EXE$GL_ERASEPPT       ********    X  02    IRP$V_MBXIO         = 0000000A
EXE$INSERTIRP         ********    X  02    IRP$V_PAGIO         = 00000002
EXE$INSIOQ            ********    X  02    IRP$V_PHYSIO        = 00000008
EXE$PROBEW            ********    X  02    IRP$V_SWAPIO        = 00000006
EXE$QXQPPKT           ********    X  02    IRP$V_TERMIO        = 00000009
IO$V_ERASE          = 0000000A             IRP$V_VIRTUAL       = 00000004
IO$_DSE             = 00000015             IRP$W_BOFF          = 00000030
IO$_WRITELBLK       = 00000020             IRP$W_CHAN          = 00000028
IO$_WRITEPBLK       = 0000000B             IRP$W_FUNC          = 00000020
IO$_WRITEVBLK       = 00000030             IRP$W_SIZE          = 00000008
IOC$BUFPOST           0000012B RG    02    IRP$W_STS           = 0000002A
IOC$CVTLOGPHY         ********    X  02    IRPE$L_BCNT1        = 00000034
IOC$DIRPOST1          000006AF RG    02    IRPE$L_BCNT2        = 00000040
IOC$GL_PSBL           ********    X  02    IRPE$L_EXTEND       = 00000054
IOC$GL_PSFL           ********    X  02    IRPE$L_SVAPTE1      = 0000002C
IOC$IOPOST            00000004 RG    02    IRPE$L_SVAPTE2      = 00000038
IOC$MAPVBLK           ********    X  02    IRPE$V_EXTEND       = 0000000B
IOC$QNXTSEG           00000478 RG    02    IRPE$W_BOFF1        = 00000030
IOC$QNXTSEG1          00000484 RG    02    IRPE$W_BOFF2        = 0000003C
IOC$QTOACP            00000510 R     02    IRPE$W_STS          = 0000002A
IOC$WAKACP            00000521 RG    02    JIB$L_BYTCNT        = 00000020
IOPOST                0000000D R     02    MMG$DECPHDREF1        ********    X  02
IPL$_IOPOST         = 00000004             MMG$GL_MAXPFN         ********    X  02
IPL$_SYNCH          = 00000008             MMG$GL_SYSPHD         ********    X  02
IRP$B_EFN           = 00000022             MMG$INSPFNT           ********    X  02
IRP$B_RMOD          = 0000000B             MMG$REFCNTNEG         ********    X  02
IRP$C_LENGTH        = 000000C4             MMG$RELPFN            ********    X  02
IRP$L_ABCNT         = 00000040             MMG$SUBSECREF         ********    X  02
```

```
MMG$UNLOCK                 ********  X    02
MMG$UPDSECAST              ********  X    02
MOVBUF                     0000074C R     02
NOTACP                     00000136 R     02
NOTFCPWCB                  0000056B R     02
PAGIO                      00000194 R     02
PAGIO_DONE                 00000221 R     02
PAGIO_DONE1                00000242 R     02
PAGIO_DONE2                00000249 R     02
PAGIO_ERR                  000002EC R     02
PAGIO_ERR_DONE             00000274 R     02
PAGIO_KAST                 00000280 R     02
PAGIO_OR_SWAPIO            000000CA R     02
PAGRD_DONE                 000001DF R     02
PAGRD_ERR                  000002A0 R     02
PAGWRT_ERR                 000001DC R     02
PAGWRT_ERR1                0000032F R     02
PAGWRT_ERR_DONE            000001D8 R     02
PCB$L_JIB                = 00000080
PCB$L_PHD                = 0000006C
PCB$W_BIOCNT             = 0000003A
PCB$W_DIOCNT             = 0000003E
PFN$AB_STATE               ********  X    02
PFN$AB_TYPE                ********  X    02
PFN$AL_BAK                 ********  X    02
PFN$AW_REFCNT              ********  X    02
PFN$AW_SWPVBN              ********  X    02
PFN$C_ACTIVE             = 00000007
PFN$C_BADPAGLST          = 00000002
PFN$C_PPGTBL             = 00000004
PFN$C_RDERR              = 00000004
PFN$M_COLLISION          = 00000010
PFN$M_DELCON             = 00000010
PFN$M_PAGTYP             = 00000007
PFN$V_COLLISION          = 00000004
PFN$V_GBLBAK             = 00000017
PFN$V_MODIFY             = 00000007
PFN_IO_DONE                00000384 R     02
PHD$L_BIOCNT             = 00000054
PHD$W_PHVINDEX           = 00000042
PMS$END_RQ                 ********  X    02
PMS$GL_DIRIO               ********  X    02
PMS$GL_SPLIT               ********  X    02
PMS$START_RQ               ********  X    02
PR$_IPL                  = 00000012
PR$_SIRR                 = 00000014
PRI$_IOCOM               = 00000001
PRI$_NULL                = 00000000
PRI$_TICOM               = 00000004
PRI$_TOCOM               = 00000003
PRITBL                     00000000 R     02
PTE$M_OWN                = 01800000
PTE$M_PROT               = 78000000
PTE$S_PFN                = 00000015
PTE$V_PFN                = 00000000
PTE$V_VALID              = 0000001F
QNXTSEG                    00000414 R     02
RSN$_ASTWAIT             = 00000001
RSN$_MAILBOX             = 00000002
SCH$GL_PCBVEC              ********  X    02
SCH$GQ_COLPGWQ             ********  X    02
SCH$POSTEF                 ********  X    02
SCH$QAST                   ********  X    02
SCH$RAVAIL                 ********  X    02
SCH$RSE                    ********  X    02
SCH$WAKE                   ********  X    02
SS$_ACCVIO               = 0000000C
SS$_ILLBLKNUM            = 000000DC
SS$_INCSEGTRA            = 00002234
TMP...                   = 00000000
UCB$B_DEVCLASS           = 00000040
UCB$L_DEVCHAR            = 00000038
UCB$L_MAXBCNT            = 000000B4
UCB$L_MAXBLOCK           = 000000B0
UCB$L_VCB                = 00000034
UCB$W_QLEN               = 0000006A
UNLK                       0000079D R     02
UNLOCK                     00000096 R     02
UNLOCK_MORE                0000076E R     02
VA$M_BYTE                = 000001FF
VA$S_BYTE                = 00000009
VA$V_VPN                 = 00000009
VCB$L_AQB                = 00000010
VIRTUAL_LOGIO              000003B5 R     02
WCB$B_ACCESS             = 0000000B
WCB$V_NOTFCP             = 00000002
WQH$W_WQCNT              = 00000008
XQP                        0000054B R     02
```

```
                              +----------------+
                              ! Psect synopsis !
                              +----------------+

PSECT name                Allocation          PSECT No.  Attributes
----------                ----------          ---------  ----------
.  ABS  .                 00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   REL?
```

PSECT name	Allocation	PSECT No.	Attributes									
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
$AEXENONPAGED	000007AC (1964.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:00.50
Command processing	131	00:00:00.63	00:00:04.31
Pass 1	578	00:00:24.30	00:01:06.34
Symbol table sort	0	00:00:03.89	00:00:14.34
Pass 2	261	00:00:05.25	00:00:17.41
Symbol table output	25	00:00:00.20	00:00:00.32
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1034	00:00:34.39	00:01:43.26

The working set limit was 2100 pages.
143058 bytes (280 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2514 non-local and 101 local symbols.
1399 source lines were read in Pass 1, producing 20 object records in Pass 2.
41 pages of virtual memory were used to define 40 macros.

! Macro library statistics !

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	28
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	37

2665 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:IOCIOPOST/OBJ=OBJ$:IOCIOPOST MSRC$:IOCIOPOST/UPDATE=(ENH$:IOCIOPOST)+EXECML$/LIB

0375 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FORKCNTRL
LIS

FILINIWCB
LIS

IMGDECODE
LIS

INIT
LIS

IOLOCK
LIS

IODEF
LIS

IOCIOPOST
LIS

GLOBALS
LIS

IMGMAPISD
LIS